DATABASE MANAGEMENT LAB PRACTICAL

(**Semester -IV** of B.Tech)

As per the curricullam and syllabus
of
Bharath Institute of Higher Education & Research

(DBMS Lab Manual)

Bharath
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)
ACCREDITED WITH 'A' GRADE BY NAAC

NEW EDITION

PREPARED BY

DR. M.K.VIDHYALAKSHMI

Bharath

INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Declared as Deemed-to-be University under section 3 of UGC Act, 1956)

(Vide Notification No. F.9-5/2000 - U.3, Ministry of Human Resource Development, Govt. of India, dated 4th July 2002)

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

LAB MANUAL

SUBJECT NAME: DATABASE MANAGEMENT SYSTEMS LAB

SUBJECT CODE: BCS4L1

Regulation R2015
(2015-2016)

BCS4L1	DBMS LABORATORY	L	T	P	C
	Total Contact Hours - 30	0	0	3	2
	Prerequisite –Database Management System				
	Lab Manual Designed by – Dept. of Computer Science and Engineering				

OBJECTIVES: The main objective isstudents gain knowledge about databases for storing the data and to share the data among different kinds of users for their business operations.

COURSE OUTCOMES (COs)

CO1	Develop database modeling for a problem.
CO2	Design a database using normalization.
CO3	Implement a data base query language.
CO4	Develop GUI using front end tool.
CO5	Develop a connection between frontend and database.
CO6	Implement a Data Manipulation Language.

MAPPING BETWEEN COURSE OUTCOMES & PROGRAM OUTCOMES
(3/2/1 INDICATES STRENGTH OF CORRELATION) 3- High, 2- Medium, 1-Low

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1			S									S	2	3	
CO2		2	2		3	1		2	2	1		S	2	3	
CO3		S	S	2			2					2	2	3	
CO4					2								2	3	
CO5			S						2	2		2	2	3	
CO6	2	S		1	2			2			2		2	3	

Category	Professional Core (PC)
Approval	37th Meeting of Academic Council, May 2015

LIST OF EXPERIMENTS

1. Data Definition, Manipulation of base tables and views
2. High level programming language extensions.
3. Front end tools.
4. Forms-Triggers-Menu Design.
5. Reports.
6. Database Design and implementation

An exercise using Open Source Software like MySQL

DATABASE MANAGEMENT SYSTEMS (DBMS) LAB - [BCS4L1]

LIST OF EXPERIMENTS

	NAME OF THE EXPERIMENT
1	Data definition languages (ddl), Data manipulation language (dml) commands of base tables and views
2	High level programming language extensions
3	Front end tools
4	Forms-triggers-menu design.
5	Reports
6	Design and implementation of employee
7	An exercise using Open-Source Software like MySQL

CONTENT

EX.NO:1a

DATA DEFINITION LANGUAGES (DDL) COMMANDS Of Base Tables and Views

A Data Definition Language (DDL) statement is used to define the database structure or schema.

Aim:

To study and execute the DDL commands in RDBMS.

DDL commands:

- ✴ CREATE
- ✴ ALTER
- ✴ DROP
- ✴ RENAME
- ✴ TRUNCATE

SYNTAX'S OF COMMANDS

CREATE TABLE:

To make a new database, table, index, or stored query. A create statement in SQL creates an object inside of a relational database management system (RDBMS).

```
CREATE TABLE <table_name>
(
Column_name1 data_type ([size]),
Column_name2 data_type ([size]),
.
.
.
Column_name-n data_type ([size])
);
```

ALTER A TABLE:

To modify an existing database object. Alter the structure of the database.

To add a column in a table

ALTER TABLE table_name ADD column_name datatype;

To delete a column in a table

ALTER TABLE table_name DROP column column_name;

DROP TABLE:

Delete Objects from the Database

DROP TABLE table_name;

TRUNCATE TABLE:

Remove all records from a table, including all spaces allocated for the records are removed.

TRUNCATE TABLE table_name;

EXERCISE:

Create Table

```
SQL> create table employee
  2 (
  3 empid varchar(10) primary key,
  4 empname varchar2(20) not null,
  5 gender varchar2(7) not null,
  6 age number(3) not null,
  7 dept varchar2(15) not null,
  8 dob date not null,
  9 doj date not null
 10);
```

Table created.

```
SQL> create table salary
  2  (
  3 empid varchar(10) references employee(empid),
  4 salary number(10) not null,
  5  dept varchar(15) not null,
  6  branch varchar2(20) not null
  7  );
Table created.

SQL> create table branchtable
  2  (
  3 branch varchar2(20) not null,
  4 city varchar2(20) not null
  5  );
Table created.
```

DESCRIBE TABLE

```
SQL> desc employee;
 Name                      Null?            Type
 ----------------------------------------- -------- ------------------------------

 EMPID                     NOT NULL    VARCHAR2(10)
 EMPNAME                   NOT NULL    VARCHAR2(20)
 GENDER                    NOT NULL    VARCHAR2(7)
 AGE                       NOT NULL     NUMBER(3)
 DEPT                      NOT NULL     VARCHAR2(15)
 DOB                       NOT NULL     DATE
 DOJ                       NOT NULL     DATE
```

SQL> desc salary;

Name	Null?	Type
EMPID		VARCHAR2 (10)
SALARY	NOT NULL	NUMBER (10)
DEPT	NOT NULL	VARCHAR2 (15)
BRANCH	NOT NULL	VARCHAR2 (20)

SQL> desc branchtable;

Name	Null?	Type
BRANCH	NOT NULL	VARCHAR2 (20)
CITY	NOT NULL	VARCHAR2 (20)

ALTER TABLE

I. ADD:

SQL> alter table employee add(designation varchar2(15));

Table altered.

SQL> alter table salary add(constraint nithi unique(empid));

Table altered.

II.MODIFY

SQL> alter table employee modify (designation varchar2(20));

Table altered.

RENAME TABLE

```
SQL> create table emp
  2  (
  3  empid varchar2(10),
  4  empname varchar2(20),
  5  age number(3),
  6  sex char
  7  );
```

Table created.

```
SQL> rename emp to empl;
```

Table renamed.

```
SQL> desc empl;
 Name                      Null?    Type
 ------------------------------------------------------------------
 EMPID                              VARCHAR2(10)
 EMPNAME                            VARCHAR2(20)
 AGE                                NUMBER(3)
 SEX                                CHAR(1)
```

```
SQL> desc emp;
ERROR:
ORA-04043: object emp does not exist
```

Table altered.

TRUNCATE TABLE DATA

```
SQL> insert into emp values(&no,'&name','&dept',&age,'&sex');
Enter value for no: 1
Enter value for name: arun
```

```
Enter value for dept: it
Enter value for age: 22
Enter value for sex: m
old 1: insert into emp values(&no,'&name','&dept',&age,'&sex')
new 1: insert into emp values(1,'arun','it',22,'m')
1 row created.

SQL> insert into emp values(&no,'&name','&dept',&age,'&sex');
Enter value for no: 2
Enter value for name: bala
Enter value for dept: service
Enter value for age: 26
Enter value for sex: m
old 1: insert into emp values(&no,'&name','&dept',&age,'&sex')
new 1: insert into emp values(2,'bala','service',26,'m')
1 row created.
SQL> insert into emp values(&no,'&name','&dept',&age,'&sex');
Enter value for no: 3
Enter value for name: chitra
Enter value for dept: sales
Enter value for age: 25
Enter value for sex: f
old 1: insert into emp values(&no,'&name','&dept',&age,'&sex')
new 1: insert into emp values(3,'chitra','sales',25,'f')
1 row created.
SQL> select * from emp;
    EMPID    EMPNAME      DEPT         AGE SEX
    ---------- ---------- ---------- ---------- -------- ----------
       1           arun                it              22       m
```

2	bala	service	26	m
3	chitra	sales	25	f

SQL> commit;

Commit complete.

SQL> truncate table emp;

Table truncated.

SQL> select * from emp;

no rows selected

SQL> commit;

Commit complete.

DROP TABLE

SQL> drop table empl;

Table dropped.

SQL> desc empl;

ERROR:

ORA-04043: object empl does not exist

RESULT:

Thus executed the DDL commands in RDBMS

EX.NO:1b

DATA MANIPULATION LANGUAGE (DML) OF BASE TABLES AND VIEWS

Data manipulation language allows the users to query and manipulate data in existing schema in object. It allows following data to insert, delete, update and recovery data in schema object.

Aim:

To study DML commands in RDBMS.

DML COMMANDS:

- INSERT
- UPDATE
- DELETE
- SELECT

QUERY:

Query is a statement in the DML that request the retrieval of data from database.

- The portion of the DML used in a Query is called Query language. The SELECT statement is used to query a database

SYNTAX OF COMMANDS

INSERT:

Values can be inserted into table using insert commands. There are two types of insert commands. They are multiple value insert commands (using '&' symbol) single value insert command (without using '&'symbol)

Syntax:

INSERT INTO table_name VALUES (value1, value2, value3,…..);

(OR)

INSERT INTO table_name (column1, column2, column3,….) VALUES (value1,value2,value3,…..);

UPDATE:

This allows the user to update the particular column value using the where clause condition.

Syntax:

UPDATE <table_name> SET <col1=value> WHERE <column=value>;

DELETE:

This allows you to delete the particular column values using where clause condition.

Syntax:

DELETE FROM <table_name> WHERE <condition>;

SELECT:

The select statement is used to query a database .This statement is used to retrieve the information from the database. The SELECT statement can be used in many ways. They are:

1. **Selecting some columns** :
   To select specified number of columns from the table the

   Following command is used.

   Syntax**:**

   SELECT column_name FROM table_name;

2. **Query All Columns**:

   To select all columns from the table * is used instead of column names.

   Syntax**:**

   SELECT * FROM table_name;

3. Select using DISTINCT:

The DISTINCT keyword is used to return only different values (i.e.) this command does not select the duplicate values from the table.

Syntax**:**

SELECT DISTINCT column name(s) FROM table_name;

4. Select using IN:

If you want to get the rows which contain certain values, the best way to do it is to use the IN conditional expression.

Syntax**:**

SELECT column name(s) FROM table_name WHERE

Column name IN (value1, value2,……,value-n);

5. **Select using BETWEEN**:

BETWEEN can be used to get those items that fall within a range.

Syntax**:**

SELECT column name FROM table_name WHERE

Column name BETWEEN value1 AND value2;

6. Renaming:

The select statement can be used to rename either a column or the entire table.

Syntax**:**

Renaming a column:

SELECT column name AS new name FROM table_name;

Renaming a table:

SELECT column name FROM table_name AS newname;

7. **Sorting**:

The select statement with the **order by Clause** is used to sort the contents

Table either in ascending or descending order.

Syntax**:**

SELECT column name FROM table_name WHERE

Condition ORDER BY column name ASC/DESC;

8. **To select by matching some patterns**:

The select statement along with **like clause** I is used to match strings. The **like** condition is used to specify a search pattern in a column.

Syntax**:**

SELECT column name FROM table_name WHERE Column name LIKE "% or-";

%: Matches any sub string.

- : Matches a single character.

9. SELECT INTO statement:

The SELECT INTO statement is most often used to create backup copies of tables or for archiving records.

Syntax**:**

SELECT Column_name(s) INTO variable_name(s) FROM table_name

WHERE condition.

10. **To Select NULL values**:

We can use the SELECT statement to select the 'null' values also.

For retrieving roes where some of the columns have been defined as NULLs there is a special comparison operator of the form IS [NOT]NULL.

Syntax:

SELECT column name FROM table_name WHERE Column name IS NULL;

11. **Select using AND, OR, NOT**:

We can combine one or more conditions in a SELECT statement using the logical operators AND, OR, NOT.

Syntax:

SELECT column name FROM table_name WHERE Condition1 LOGICAL OPERATOR condition2;

EXERCISE:

INSERT COMMAND

SQL> insert into employee

values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desig');

Enter value for empid: it9001

Enter value for empname: arunkumar

Enter value for gender: male

Enter value for age: 22

Enter value for dept: it

Enter value for dob: 12-jan-1988

Enter value for doj: 23-oct-2006

Enter value for desig: manager

old 1: insert into employee

values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desi

new 1: insert into employee values('it9001','arunkumar','male',22,'it','12-jan-1988','23-oct-2006'

1 row created.

SQL> insert into employee

values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desig');

Enter value for empid: it9001

Enter value for empname: arunkumar

Enter value for gender: male

Enter value for age: 22

Enter value for dept: it

Enter value for dob: 12-jan-1988

Enter value for doj: 23-oct-2006

Enter value for desig: manager

old 1: insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desi
new 1: insert into employee values('it9001','arunkumar','male',22,'it','12-jan-1988','23-oct-2006'
1 row created.

SQL> insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desig');
Enter value for empid: it9002
Enter value for empname: balakrishnan
Enter value for gender: male
Enter value for age: 27
Enter value for dept: it
Enter value for dob: 27-mar-1983
Enter value for doj: 02-dec-2008
Enter value for desig: coordinator
old 1: insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desi
new 1: insert into employee values('it9002','balakrishnan','male',27,'it','27-mar-1983','02-dec-20
1 row created.

SQL> insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desig');
Enter value for empid: acc9001
Enter value for empname: kannan
Enter value for gender: male
Enter value for age: 35
Enter value for dept: accounts
Enter value for dob: 28-dec-1975

Enter value for doj: 01-jan-1995

Enter value for desig: manager

old 1: insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desi

new 1: insert into employee values('acc9001','kannan','male',35,'accounts','28-dec-1975','01-jan-1

1 row created.

SQL> insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desig');

Enter value for empid: acc9002

Enter value for empname: magudeshwaran

Enter value for gender: male

Enter value for age: 27

Enter value for dept: accounts

Enter value for dob: 25-aug-1983

Enter value for doj: 12-apr-2000

Enter value for desig: asst manager

old 1: insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desi

new 1: insert into employee values('acc9002','magudeshwaran','male',27,'accounts','25-aug-1983','1

1 row created.

SQL> insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desig');

Enter value for empid: ser9001

Enter value for empname: jagadheesh

Enter value for gender: male

Enter value for age: 33

Enter value for dept: service

Enter value for dob: 31-mar-1877

Enter value for doj: 3-jun-1999

Enter value for desig: manager

old 1: insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desi
new 1: insert into employee values('ser9001','jagadheesh','male',33,'service','31-mar-1877','3-jun
1 row created.

SQL> insert into employee

values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desig');

Enter value for empid: ser9006

Enter value for empname: muruganandam

Enter value for gender: male

Enter value for age: 35

Enter value for dept: service

Enter value for dob: 09-aug-1975

Enter value for doj: 02-mar-2000

Enter value for desig: painter

old 1: insert into employee
values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desi
new 1: insert into employee values('ser9006','muruganandam','male',35,'service','09-aug-1975','02-
1 row created.

SQL> insert into employee

values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desig');

SQL> /

Enter value for empid: sal9001

Enter value for empname: suresh

Enter value for gender: male

Enter value for age: 40

Enter value for dept: sales

Enter value for dob: 12-jul-1970

Enter value for doj: 01-apr-1996

Enter value for desig: manager

old 1: insert into employee

values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desi

new 1: insert into employee values('sal9001','suresh','male',40,'sales','12-jul-1970','01-apr-1996

1 row created.

SQL> insert into employee

values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desig');

Enter value for empid: sal9006

Enter value for empname: sharmila

Enter value for gender: female

Enter value for age: 27

Enter value for dept: sales

Enter value for dob: 12-jan-1983

Enter value for doj: 09-aug-2007

Enter value for desig: executive

old 1: insert into employee

values('&empid','&empname','&gender',&age,'&dept','&dob','&doj','&desi

new 1: insert into employee values('sal9006','sharmila','female',27,'sales','12-jan-1983','09-aug-

1 row created.

SQL> insert into salary values(‘&empid’,&salary,’&dept’,’&branch’);

Enter value for empid: it9002

Enter value for salary: 18000

Enter value for dept: it

Enter value for branch: abt maruthi

old 1: insert into salary values('&empid',&salary,'&dept','&branch')

new 1: insert into salary values('it9002',18000,'it','abt maruthi')

1 row created.

SQL> insert into salary values(‘&empid’,&salary,’&dept’,’&branch’);

Enter value for empid: acc9001

Enter value for salary: 35000

Enter value for dept: accounts

Enter value for branch: cars india

old 1: insert into salary values('&empid',&salary,'&dept','&branch')

new 1: insert into salary values('acc9001',35000,'accounts','cars india')

1 row created.

SQL> insert into salary values(‘&empid’,&salary,’&dept’,’&branch’);

Enter value for empid: acc9002

Enter value for salary: 26000

Enter value for dept: accounts

Enter value for branch: cars india

old 1: insert into salary values('&empid',&salary,'&dept','&branch')

new 1: insert into salary values('acc9002',26000,'accounts','cars india')

1 row created.

SQL> insert into salary values(‘&empid’,&salary,’&dept’,’&branch’);

Enter value for empid: ser9001

Enter value for salary: 35000

Enter value for dept: service

Enter value for branch: chennai cars

old 1: insert into salary values('&empid',&salary,'&dept','&branch')

new 1: insert into salary values('ser9001',35000,'service','chennai cars')

1 row created.

SQL> insert into salary values(‘&empid’,&salary,’&dept’,’&branch’);

Enter value for empid: ser9006

Enter value for salary: 12000

Enter value for dept: service

Enter value for branch: greenland cars

old 1: insert into salary values('&empid',&salary,'&dept','&branch')

new 1: insert into salary values('ser9006',12000,'service','greenland cars')

1 row created.

SQL> insert into salary values(‘&empid’,&salary,’&dept’,’&branch’);

Enter value for empid: sal9001

Enter value for salary: 40000

Enter value for dept: sales

Enter value for branch: abt maruthi

old 1: insert into salary values('&empid',&salary,'&dept','&branch')

new 1: insert into salary values('sal9001',40000,'sales','abt maruthi')

1 row created.

SQL> insert into salary values(‘&empid’,&salary,’&dept’,’&branch’);

Enter value for empid: sal9006

Enter value for salary: 17000

Enter value for dept: sales

Enter value for branch: abt maruthi

old 1: insert into salary values('&empid',&salary,'&dept','&branch')

new 1: insert into salary values('sal9006',17000,'sales ','abt maruthi')

1 row created.

SQL> select * from salary;

EMPID	SALARY	DEPT	BRANCH
it9001	35000	it	abt maruthi
it9002	18000	it	abt maruthi
acc9001	35000	accounts	cars india
acc9002	26000	accounts	cars india
ser9001	35000	service	chennai cars
ser9006	12000	service	greenland cars
sal9001	40000	sales	abt maruthi
sal9006	17000	sales	abt maruthi

8 rows selected.

SQL> select * from employee;

EMPID EMPNAME GENDER AGE DEPT DOB

DOJ DESIGNATION

---

it9001 arunkumar male 22 it 12-JAN-88

23-OCT-06 manager

it9002 balakrishnan male 27 it 27-MAR-83

02-DEC-08 coordinator

acc9001 kannan male 35 accounts 28-DEC-75

01-JAN-95 manager

EMPID EMPNAME GENDER AGE DEPT DOB

---------- ---------- -------- ---------- ---------- ----------

DOJ DESIGNATION

---------- ----------

acc9002 magudeshwaran male 27 accounts 25-AUG-83

12-APR-00 asst manager

ser9001 jagadheesh male 33 service 31-MAR-77

03-JUN-99 manager

ser9006 muruganandam male 35 service 09-AUG-75

02-MAR-00 painter

EMPID EMPNAME GENDER AGE DEPT DOB

---------- ---------- -------- ---------- ---------- ----------

DOJ DESIGNATION

---------- ----------

sal9001 suresh male 40 sales 12-JUL-70

01-APR-96 manager

sal9006 sharmila female 27 sales 12-JAN-83

09-AUG-07 executive

8 rows selected.

SQL> insert into branchtable values('&branch','&city');

Enter value for branch: abt maruthi

Enter value for city: chennai

old 1: insert into branchtable values('&branch','&city')

new 1: insert into branchtable values('abt maruthi','chennai')

1 row created.

SQL> select * from salary;

EMPID SALARY DEPT BRANCH

---------- ---------- ---------- ----------

it9001	35000	it	abt maruthi
it9002	18000	it	abt maruthi
acc9001	35000	accounts	cars india
acc9002	26000	accounts	cars india
ser9001	35000	service	chennai cars
ser9006	12000	service	greenland cars
sal9001	40000	sales	abt maruthi
sal9006	17000	sales	abt maruthi

8 rows selected.

SQL> insert into branchtable values('&branch','&city');

Enter value for branch: cars india

Enter value for city: vellore

old 1: insert into branchtable values('&branch','&city')

new 1: insert into branchtable values('cars india','vellore')

1 row created.

SQL> insert into branchtable values('&branch','&city');

Enter value for branch: chennai cars

Enter value for city: thambaram

old 1: insert into branchtable values('&branch','&city')

new 1: insert into branchtable values('chennai cars','thambaram')

1 row created.

SQL> insert into branchtable values('&branch','&city');

Enter value for branch: greenland cars

Enter value for city: kanchipuram

old 1: insert into branchtable values('&branch','&city')

new 1: insert into branchtable values('greenland cars','kanchipuram')

1 row created.

SQL> select * from branchtable;

BRANCH	CITY
abt maruthi	chennai
cars india	vellore
chennai cars	thambaram
greenland cars	kanchipuram

UPDATE COMMAND

SQL> update employee set empname = 'arunprasanth' where empid='it9001';

1 row updated.

SQL> update employee set designation='&designation' where empname='&empname';

Enter value for designation: supervisor

Enter value for empname: muruganandam

old 1: update employee set designation='&designation' where empname='&empname'

new 1: update employee set designation='supervisor' where empname='muruganandam'

1 row updated.

SQL> select empname,designation from employee;

EMPNAME	DESIGNATION
arunprasanth	manager
balakrishnan	coordinator
kannan	manager
magudeshwaran	asst manager
jagadheesh	manager

muruganandam supervisor

suresh manager

sharmila executive

8 rows selected.

SELECT COMMAND

To retrieve particular column

SQL> select empname from emp;

EMPNAME

---

arun

bala

bakyaraj

chitra

To retrieve all columns

SQL> select * from emp;

EMPID	EMPNAME	DEPT	AGE	S
1	arun	it	22	m
2	bala	accounts	26	m
3	bakyaraj	stores	30	m
4	chitra	sales	24	f

DELETE COMMAND

To delete particular record

SQL> delete emp where empid=1;

1 row deleted.

SQL> select * from emp;

EMPID	EMPNAME	DEPT	AGE	S
2	bala	accounts	26	m
3	bakyaraj	stores	30	m
4	chitra	sales	24	f

To delete all records

SQL> delete from emp;

3 rows deleted.

SQL> create table student (idno number, name varchar(10),branch varchar(4));

Table created.

SQL> desc student;

NAME	NULL?	TYPE
IDNO		NUMBER
NAME		VARCHAR2(10)
BRANCH		VARCHAR2(4)

SQL> alter table student add degree varchar(10);

Table altered.

SQL> desc student;

NAME	NULL?	TYPE
IDNO		NUMBER

NAME VARCHAR2 (10)

BRANCH VARCHAR2 (4)

DEGREE VARCHAR2 (10)

SQL> alter table student modify degree

varchar(6); Table altered.

SQL> desc student;

NAME NULL? TYPE

IDNO NUMBER

NAME VARCHAR2 (10)

BRANCH VARCHAR2 (4)

DEGREE VARCHAR2 (6)

SQL> insert into student (name, degree, branch, idno) values('ASHOK','BE','CSE',01);

1 row created.

SQL> insert into student values(02,'BHAVANA','CSE','BE');

1 row created.

SQL> insert into student values(&idno, &name, &branch, &degree);

Enter value for idno: 03

Enter value for name: 'CAVIN'

Enter value for branch: 'CSE'

Enter value for degree: 'BE'

old 1: insert into student values(&idno,&name,&branch,&degree)

new 1: insert into student values(03,'CAVIN','CSE','BE')

1 row created.

Enter value for idno: 04

Enter value for name: 'DANNY'

Enter value for branch: 'IT'

Enter value for degree: 'BE'

old 1: insert into student values(&idno,&name,&branch,&degree)

new 1: insert into student values(04,'DANNY','IT','BE')

1 row created.

SQL> /

Enter value for idno: 05

Enter value for name: 'HARRY'

Enter value for branch: 'IT'

Enter value for degree: 'BE'

old 1: insert into student values(&idno,&name,&branch,&degree)

new 1: insert into student values(05,'HARRY','IT','BE')

1 row created.

SQL> select * from student;

IDNO	NAME	BRAN	DEGREE
1	ASHOK	CSE	BE
2	BHAVANA	CSE	BE
3	CAVIN	CSE	BE
4	DANNY	IT	BE
5	HARRY	IT	BE

SQL> update student set degree='B.TECH' where branch='IT';

2 rows updated.

SQL> select * from student;

IDNO	NAME	BRAN	DEGREE
1	ASHOK	CSE	BE
2	BHAVANA	CSE	BE
3	CAVIN	CSE	BE
4	DANNY	IT	B.TECH
5	HARRY	IT	B.TECH

SQL> delete from student where idno=5;

1 row deleted.

CREATING TABLES WITH CONSTRAINTS:

NOT NULL

SQL> select * from student;

IDNO	NAME	BRAN	DEGREE
1	ASHOK	CSE	BE
2	BHAVANA	CSE	BE
3	CAVIN	CSE	BE
4	DANNY	IT	B.TECH

SQL> create table staff

(

idno number (4) not null,name

varchar(10),branch varchar(6)

); Table created.

SQL> desc staff;

NAME	NULL?	TYPE
IDNO	NOT NULL	NUMBER(4)
NAME		VARCHAR2(10)
BRANCH		VARCHAR2(6)

SQL> insert into staff values (&idno, &name, &branch);

Enter value for idno: 1

Enter value for name: 'ABILASH'

Enter value for branch: 'CSE'

old 1: insert into staff values(&idno, &name, &branch)

new 1: insert into staff values(1,'ABILASH','CSE')

1 row created.

SQL> /

Enter value for idno: 2

Enter value for name: 'ANTON'

Enter value for branch: 'CSE'

old 1: insert into staff values(&idno, &name, &branch)

new 1: insert into staff values(2,'ANTON','CSE')

1 row created.

SQL> /

Enter value for idno:

Enter value for name: 'BENNY'

Enter value for branch: 'IT'

old 1: insert into staff values(&idno,&name,&branch)

new 1: insert into staff values(,'BENNY','IT')

insert into staff values(,'BENNY','IT') *

ERROR at line 1:

ORA-00936: missing expression

UNIQUE

SQL> create table employee

(

rollno number unique,

name varchar(10),

salary number

);

Table created.

SQL> desc employee;

NAME	NULL?	TYPE
ROLLNO		NUMBER
NAME		VARCHAR2(10)
SALARY		NUMBER

SQL> insert into employee values(&rollno,&name,&salary);

Enter value for rollno: 1

Enter value for name: 'anton'

Enter value for salary: 10290

old 1: insert into employee values(&rollno,&name,&salary)

new 1: insert into employee values(1,'anton',10290)

1 row created.

SQL> /

Enter value for rollno: 2

Enter value for name: 'dharun'

Enter value for salary: 23322

old 1: insert into employee values(&rollno,&name,&salary)

new 1: insert into employee values(2,'dharun',23322)

1 row created.

SQL> /

Enter value for rollno: 1

Enter value for name: 'aaron'

Enter value for salary: 32212

old 1: insert into employee values(&rollno,&name,&salary)

new 1: insert into employee values(1,'aaron',32212)

insert into employee values(1,'aaron',32212)

*

ERROR at line 1:

ORA-00001: unique constraint (SCOTT.SYS_C001265) violated

PRIMARY KEY

SQL> create table cars

(model number primary key,

name varchar(10),

cost number(6)

);

Table created.

SQL> desc cars;

NAME	NULL?	TYPE
MODEL	NOT NULL	NUMBER
NAME		VARCHAR2(10)
COST		NUMBER(6)

SQL> insert into cars values(&model,&name,&cost);

Enter value for model: 1098

Enter value for name: 'omni'

Enter value for cost: 200000

old 1: insert into cars values(&model,&name,&cost)

new 1: insert into cars values(1098,'omni',200000)

1 row created.

SQL> /

Enter value for model: 9087

Enter value for name: 'qualis'

Enter value for cost: 500000

old 1: insert into cars values(&model,&name,&cost)

new 1: insert into cars values(9087,'qualis',500000)

1 row created.

SQL> /

Enter value for model: 1098

Enter value for name: 'innova'

Enter value for cost: 600000

old 1: insert into cars values(&model,&name,&cost)

insert into cars values(1098,'innova',600000)

*

ERROR at line 1:

ORA-00001: unique constraint (SCOTT.SYS_C001266) violated

CHECK CONSTRAINT:

SQL> create table employ

(

rno number(5),

name varchar(10),

salary number(10) constraint no_ck check(salary between 10000 and 30000)

);

Table created.

SQL> desc employ;

NAME	NULL?	TYPE
RNO		NUMBER(5)
NAME		VARCHAR2(10)
SALARY		NUMBER(10)

SQL> insert into employ values(&rno,&name,&salary);

Enter value for rno: 1

Enter value for name: 'sachin'

Enter value for salary: 29000

old 1: insert into employ values(&rno,&name,&salary)

new 1: insert into employ values(1,'sachin',29000)

SQL> /

Enter value for rno: 20

Enter value for name: 'rohit'

Enter value for salary: 10000

old 1: insert into employ values(&rno, &name, &salary)

new 1: insert into employ values(20,'rohit',10000)

1 row created.

SQL> /

Enter value for rno: 15

Enter value for name: 'dhoni'

Enter value for salary: 40000

old 1: insert into employ values(&rno,&name,&salary)

new 1: insert into employ values(15,'dhoni',40000)

insert into employ values(15,'dhoni',40000)

*

ERROR at line 1:

ORA-02290: check constraint (SCOTT.NO_CK) violated

FOREIGN KEY

SQL> create table admin

(

stuid number constraint stuid_pk primary key,

name varchar(10),

permit number(6)

);

Table created.

```
SQL> desc admin;
 NAME                      NULL?    TYPE
 ----------------------------------------------------------------------------------------

 STUID                     NOT NULL NUMBER
 NAME                               VARCHAR2(10)
 PERMIT                             NUMBER(6)

SQL> insert into admin values(&stuid, '&name', &permit);
Enter value for stuid: 1
Enter value for name: ASWIN
Enter value for permit: 80
old 1: insert into admin values(&stuid,'&name',&permit)
new 1: insert into admin values(1,'ASWIN',80)
1 row created.
SQL> /
Enter value for stuid: 2
Enter value for name: ROHIT
Enter value for permit: 67
old 1: insert into admin values(&stuid,'&name',&permit)
new 1: insert into admin values(2,'ROHIT',67)
1 row created.

SQL> /
Enter value for stuid: 4
Enter value for name: SANJAY
Enter value for permit: 45
old 1: insert into admin values(&stuid,'&name',&permit)
```

new 1: insert into admin values(4,'SANJAY',45)

1 row created.

SQL> /

Enter value for stuid: 5

Enter value for name: KAMALINI

Enter value for permit: 35

old 1: insert into admin values(&stuid,'&name',&permit)

new 1: insert into admin values(5,'KAMALINI',35)

1 row created.

SQL> select * from admin;

STUID	NAME	PERMIT
1	ASWIN	80
2	ROHIT	67
4	SANJAY	45
5	KAMALINI	35

SQL> create table course

(

stuid number constraint sid_fk references admin(stuid),

branch varchar(6),

sec varchar(2)

);

Table created.

```
SQL> insert into course values(&stuid,'&branch','&sec');
Enter value for stuid: 1
Enter value for branch: CSE
Enter value for sec: A
old 1: insert into course values(&stuid,'&branch','&sec')
new 1: insert into course values(1,'CSE','A')
1 row created.

SQL> /
Enter value for stuid: 2
Enter value for branch: CSE
Enter value for sec: A
old 1: insert into course values(&stuid,'&branch','&sec')
new 1: insert into course values(2,'CSE','A')
1 row created.

SQL> /
Enter value for stuid: 4
Enter value for branch: IT
Enter value for sec: A
old 1: insert into course values(&stuid,'&branch','&sec')
new 1: insert into course values(4,'IT','A')
1 row created.

SQL> /
Enter value for stuid: 6
Enter value for branch: CSE
```

Enter value for sec: A

old 1: insert into course values(&stuid,'&branch','&sec')

new 1: insert into course values(6,'CSE','A')

insert into course values(6,'CSE','A')

*

ERROR at line 1:

ORA-02291: integrity constraint (SCOTT.SID_FK) violated - parent key not found

SQL> delete from admin where stuid=5;

1 row deleted.

SQL> delete from admin where stuid=1;

delete from admin where stuid=1

*

ERROR at line 1:

ORA-02292: integrity constraint (SCOTT.SID_FK) violated - child record found

SQL> select * from admin;

STUID	NAME	PERMIT
1	ASWIN	80
2	ROHIT	67
4	SANJAY	45

SQL> select * from course;

STUID	BRANCH	SE
1	CSE	A
2	CSE	A
4	IT	A

```
SQL> create table student
 (
idno varchar(4),
name varchar(10),
dept varchar(4),
degree varchar(6),
year number(4)
);
table created.

SQL> desc student;
 NAME                      NULL?    TYPE
 ----------------------------------------------------------------------------------------

 IDNO                              VARCHAR2(4)
 NAME                              VARCHAR2(10)
 DEPT                              VARCHAR2(4)
 DEGREE                            VARCHAR2(6)
 YEAR                              NUMBER(4)


SQL> insert into student values('&idno', '&name', '&dept', '&degree', &year);
Enter value for idno: A01
Enter value for name: AARON
Enter value for dept: CSE
Enter value for degree: BE
Enter value for year: 2
old 1: insert into student values('&idno','&name','&dept','&degree',&year)
```

new 1: insert into student values('a01','aaron','cse','BE',2)

1 row created.

SQL> /

Enter value for idno: A02

Enter value for name: AKIL

Enter value for dept: ECE

Enter value for degree: BE

Enter value for year: 2

old 1: insert into student values('&idno','&name','&dept','&degree',&year)

new 1: insert into student values('A02','AKIL','ECE','BE',2)

1 row created.

SQL> /

Enter value for idno: A03

Enter value for name: BENNY

Enter value for dept: IT

Enter value for degree: B.TECH

Enter value for year: 2

old 1: insert into student values('&idno','&name','&dept','&degree',&year)

new 1: insert into student values('A03','BENNY','IT','B.TECH',2)

1 row created.

SQL> /

Enter value for idno: B01

Enter value for name: COOK

Enter value for dept: CSE

Enter value for degree: BE

Enter value for year: 1

old 1: insert into student values('&idno','&name','&dept','&degree',&year)

new 1: insert into student values('B01','COOK','CSE','BE',1)

1 row created.

SQL> /

Enter value for idno: B02

Enter value for name: DANNY

Enter value for dept: MECH

Enter value for degree: BE

Enter value for year: 1

old 1: insert into student values('&idno','&name','&dept','&degree',&year)

new 1: insert into student values('B02','DANNY','MECH','BE',1)

1 row created.

SQL> /

Enter value for idno: B03

Enter value for name: ELAN

Enter value for dept: IT

Enter value for degree: B.TECH

Enter value for year: 1

old 1: insert into student values('&idno','&name','&dept','&degree',&year)

new 1: insert into student values('B03','ELAN','IT','B.TECH',1)

1 row created.

SQL> SELECT * FROM STUDENT;

IDNO	NAME	DEPT	DEGREE	YEAR
A01	AARON	CSE	BE	2
A02	AKIL	ECE	BE	2
A03	BENNY	IT	B.TECH	2
B01	COOK	CSE	BE	1
B02	DANNY	MECH	BE	1
B03	ELAN	IT	B.TECH	1

6 rows selected.

DISTINCT

SQL> select distinct dept from student;

DEPT

CSE

ECE

IT

MECH

SQL> select name from student;

NAME

AARON

AKIL

BENNY

COOK

DANNY

ELAN

6 rows selected.

IN

SQL> select * from student where year IN 2;

IDNO	NAME	DEPT	DEGREE	YEAR
A01	AARON	CSE	BE	2
A02	AKIL	ECE	BE	2
A03	BENNY	IT	B.TECH	2

SQL> select * from student where name BETWEEN 'AARON' and 'COOK';

IDNO	NAME	DEPT	DEGREE	YEAR
A01	AARON	CSE	BE	2
A02	AKIL	ECE	BE	2
A03	BENNY	IT	B.TECH	2
B01	COOK	CSE	BE	1

AS

SQL> select IDNO as rollno from student;

ROLLNO

A01

A02

A03

B01

B02

B03

6 rows selected.

SORT

SQL> select * from student where year<3 order by name desc;

IDNO	NAME	DEPT	DEGREE	YEAR

B03	ELAN	IT	B.TECH	1
B02	DANNY	MECH	BE	1
B01	COOK	CSE	BE	1
A03	BENNY	IT	B.TECH	2
A02	AKIL	ECE	BE	2
A01	AARON	CSE	BE	2

6 rows selected.

SQL> select * from student where year<3 order by dept asc;

IDNO	NAME	DEPT	DEGREE	YEAR
A01	AARON	CSE	BE	2
B01	COOK	CSE	BE	1
A02	AKIL	ECE	BE	2
A03	BENNY	IT	B.TECH	2
B03	ELAN	IT	B.TECH	1
B02	DANNY	MECH	BE	1

6 rows selected.

LIKE

SQL> select * from student where name LIKE '%Y';

IDNO	NAME	DEPT	DEGREE	YEAR
A03	BENNY	IT	B.TECH	2
B02	DANNY	MECH	BE	1

SQL> select * from student where name LIKE 'A%';

IDNO	NAME	DEPT	DEGREE	YEAR
A01	AARON	CSE	BE	2
A02	AKIL	ECE	BE	2

IS NULL

SQL> select * from student where IDNO IS NULL;

no rows selected

LOGICAL OR

SQL> select * from student where IDNO='A01' OR IDNO='B01';

IDNO	NAME	DEPT	DEGREE	YEAR
A01	AARON	CSE	BE	2
B01	COOK	CSE	BE	1

RESULT:

Thus the data manipulation language (dml) of base tables and views are executed.

EX.NO: 2

HIGH LEVEL PROGRAMMING LANGUAGE EXTENSIONS

Aim:

To implement PL/SQL program using control structures, procedures and functions.

(a) CONTROL STRUCTURE:

Introduction:

An interactive control statement is used when we want to repeat the execution of one or more statements for specified number of times.

If-then:

The simplest way of IF statement associates a condition with a sequence of statements enclosed by the keywords THEN and END IF as follows.

Syntax:

```
IF condition THEN
        Sequence_of_statements
END IF;
```

The sequence of statements is executed only if the condition is true. If the condition is false or null, then if statement does nothing. The control passes to the next statement.

If-then-else:

The second form of IF statement adds the keyword ELSE followed by alternative sequence of statements, as follows

Syntax:

```
IF condition THEN
        Sequence_of_statements1
ELSE
        Sequence_of_statements2
ENDIF;
```

The sequence of statements in the ELSE clause is executed only if the condition is false or null. Thus the ELSE clause ensures that a sequence of statements is executed.

If-then-elseif:

Sometimes you want to select from several mutually exclusive alternatives. The third form of IF statement uses ELSEIF to introduce additional as follows

Syntax:

```
IF conditional1 THEN
        Sequence_of_statements1
ELSEIF condition2 THEN
        Sequence_of_statements2
ELSE
        Sequence_of_statements3
ENDIF;
```

Nested If:

Syntax:

```
IF condition THEN
        statement1;
ELSE
        IF condition THEN
                statement2;
        ELSE
                statement3;
        END
IF; END IF;
```

Case statement:

Like the IF statement, the CASE statement selects one of statements to execute. However, to select the sequence the case statement uses a selector rather than multiple Boolean expressions.

The CASE statement has the following form

Syntax:

```
CASE selector
```

```
        WHEN expression1 THEN sequence_of_statements1;
        WHEN expression2 THEN sequence_of_statements2;
                ……
        WHEN expression THEN sequence_of_statementsN;
        [ELSE sequence_of_statementsN+1;]
    END CASE;
```

Simple Loop:

The simplest form of loop statements is the basic loop which encloses a sequence of statements between the keyword LOOP and ENDLOOP as follows

Syntax:

```
    LOOP
        Sequence_of_statements
    EXIT [WHEN Condition]
    END LOOP;
```

With each iteration of the loop the sequence of the statements is executed then the control resumes at the top of the loop. If further processing is undesirable or impossibly you can use an EXIT statement to complete the loop.

While loop:

The while loop statement associates a condition with a sequence of statements enclosed by the keywords LOOP and END LOOP as follows

Syntax:

```
    WHILE condition LOOP
                Sequence_of_statements
    END LOOP;
```

Before each iteration of the loop the condition is evaluated. If the condition is true the sequence of statements is executed then the control resumes at the loop. If the condition is false or null the loop is bypassed and the control passes to the next statement.

For loop:

The number of iterations through FOR loop is known before the loop is entered. FOR loops iterate over a specified range of integers, the range is part of an iteration scheme, which is enclosed by the keywords FOR and LOOP. A double dot (..) serves as the range operator.

Syntax:

```
FOR counter IN [REVERSE]
        LowerBound..UpperBound LOOP
        Sequence_of_statements
END LOOP;
```

GOTO statement:

The GOTO statement branches to a label unconditionally. The label must be unique within its scope and must proceed an executable statement or a pl/sql block.

When executed the GOTO statement transfers control to the labeled statement or block.

Syntax:

```
Begin
  …
    GOTO insert_row;
  …
    INSERT INTO values
END;
```

NULL statement:

The null statement does nothing other than pass control to the next statement. In a conditional construct the NULL statement tells readers that a possibility has been considered, but no action is necessary.

(b) **PROCEDURES**:

- A procedure is a block that can take parameters (sometimes referred to as arguments) and be invoked.
- Procedures promote reusability and maintainability. Once validated, they can be used in number of applications.
- **A procedure has two parts**:
  1. The specification
  2. The body.

The Specification:

- The procedure specification begins with the keyword PROCEDURE and ends with the Procedure_Name or a Parameter_List.
- Parameter declarations are optional. Procedures that take no parameters are written without parentheses.

The Body:

- The procedure body begins with the keyword IS (or AS) and ends with the keyword END followed by an optimal procedure name.
- The procedure body has three parts:
  1. A Declarative part.
  2. An Executable part.
  3. An Exception-handling part (Optional).
- The declarative part contains local declarations, which are placed between the keywords IS and BEGINS.
- The keyword DECLARES, which introduces declarations in an anonymous PL/SQL block, is not used.
- The executable part contains statements, which are placed between the keywords BEGIN, and EXCEPTION (or END).

Syntax:

```
CREATE [OR REPLACE] PROCEDURE Procedure_Name [(parameter, parameter)]
IS
        [declaration_section]
BEGIN
        Executable_section
```

[EXCEPTION

Exception_section]

END [Procedure_Name];

(c) **FUNCTIONS**:

- A function is a program that might perform an action and does return a value. The function is a subprogram that computes a value.
- Like a procedure, a function has two parts:
  1. The specification
  2. The body

The Specification:

- The function specification begins with the keyword FUNCTION and ends with the RETURN clause, which specifies the data type of the return value.
- Parameter declaration are optional. Functions that take no parameters are written without parentheses.

The Body:

- The function body begins with the keyword IS (or AS) and ends with keyword END followed by an optional function name.
- The function body has three parts:

1. A Declarative part.
2. An Executable part.
3. An Exception-handling part (Optional).

- The declarative par contains local declarations, which are placed between the keywords IS and BEGIN.
- The keyword DECLARES is not used.
- The executable part contains statements, which are placed between the keywords BEGIN, and EXCEPTION (or END).

Syntax:

```
CREATE [OR REPLACE] FUNCTION function_name [(parameter [, parameter])]
        AS
        [declaration_section]
BEGIN
        executable_section
RETURN
      END [function_name];
```

EXERCISE:

FACTORIAL PROGRAM

```
SQL> declare
  2  n number(2);
  3  p number(5);
  4  i number(2);
  5  begin
  6  n:=&n;
  7  p:=1;
  8 for i in 1..n
  9  loop
 10  p:=p*i;
 11  end loop;
 12  dbms_output.put_line('Factorial value is '||to_char(p));
 13  end;
 14  /
Enter value for n: 5
old 6: n:=&n;
new 6: n:=5;
PL/SQL procedure successfully completed.

SQL> set serveroutput on;
SQL> /
Enter value for n: 5
old 6: n:=&n;
```

new 6: n:=5;

Factorial value is 120

PL/SQL procedure successfully completed.

(b)PROCEDURES

SQL> create table stud(rno number(2),mark1 number(3),mark2 number(3),total number(3),primary key(rno));

Table created.

SQL> desc stud;

Name	Null?	Type
RNO	NOT NULL	NUMBER(2)
MARK1		NUMBER(3)
MARK2		NUMBER(3)
TOTAL		NUMBER(3)

SQL> select * from stud;

RNO	MARK1	MARK2	TOTAL
1	80	85	0
2	75	84	0
3	65	80	0
4	90	85	0

```
SQL> create or replace procedure stud (rnum number) is
 2 m1 number;
 3  m2 number;
 4  total number;
 5  begin
 6  select mark1,mark2 into m1,m2 from stud where rno=rnum;
 7 if m1<m2 then
 8 update stud set total=m1+m2 where rno=rnum;
 9 end if;
```

```
 10  end;
 11  /
Procedure created.
SQL> exec studd(1);
PL/SQL procedure successfully completed.

SQL> select * from stud;
      RNO      MARK1      MARK2      TOTAL
      ---------- ----------  ---------- ----------
       1         80          85        165
       2         75          84         0
       3         65          80         0
       4         90          85         0

SQL> exec studd(4);
PL/SQL procedure successfully completed.
SQL> select * from stud;
      RNO      MARK1      MARK2      TOTAL
      ------------------------------------------------------------
       1         80          85        165
       2         75          84         0
       3         65          80         0
       4         90          85         0

SQL> exec studd(2);
PL/SQL procedure successfully completed.

SQL> exec studd(3);
PL/SQL procedure successfully completed.

SQL> select * from stud;
```

RNO	MARK1	MARK2	TOTAL
1	80	85	165
2	75	84	159
3	65	80	145
4	90	85	0

(c)FUNCTION:

```
SQL> create table stud
  2  (
  3  rno number(5),
  4  mark1 number(5),
  5  mark2 number(5),
  6  total number(5),primary key(rno)
  7  );
Table created.
SQL> desc stud;
 Name                      Null?            Type
 ----------------------------------------- -------- ----------------------------
 RNO                       NOT NULL         NUMBER(5)
 MARK1                                      NUMBER(5)
 MARK2                                      NUMBER(5)
 TOTAL                                      NUMBER(5)

SQL> insert into stud values(&rno,&mark1,&mark2,&total);
Enter value for rno: 1
Enter value for mark1: 80
Enter value for mark2: 65
Enter value for total: 0
old 1: insert into stud values(&rno,&mark1,&mark2,&total)
new 1: insert into stud values(1,80,65,0)
1 row created.
```

```
SQL> insert into stud values(&rno,&mark1,&mark2,&total);
Enter value for rno: 2
Enter value for mark1: 77
Enter value for mark2: 56
Enter value for total: 0
old 1: insert into stud values(&rno,&mark1,&mark2,&total)
new 1: insert into stud values(2,77,56,0)
1 row created.

SQL> insert into stud values(&rno,&mark1,&mark2,&total);
Enter value for rno: 3
Enter value for mark1: 89
Enter value for mark2: 90
Enter value for total: 0
old 1: insert into stud values(&rno,&mark1,&mark2,&total)
new 1: insert into stud values(3,89,90,0)
1 row created.
SQL> select * from stud;

      RNO      MARK1      MARK2      TOTAL
---------- ---------- ---------- ----------
         1         80         65          0
         2         77         56          0
         3         89         90          0

SQL> create or replace function sfunc(rnum number) return number is
  2 total stud0.total%type;
  3  m1 stud0.mark1%type;
  4  m2 stud0.mark2%type;
  5  begin
  6  select mark1,mark2 into m1,m2 from stud0 where rno=rnum;
  7  total:=m1+m2;
  8  return total;
```

```
 9  end;
10  /
```

Function created.

```
SQL> select sfunc(1) from dual;
  SFUNC(1)
-----------

       145

SQL> select sfunc(2) from dual;
  SFUNC(2)
-----------

       133

SQL> select sfunc(3) from dual;
  SFUNC(3)
-----------

       179
```

RESULT:

Thus executed high level programming language extensions

EX.NO:3

FRONT END TOOLS

Aim: Basic Study of VB Front end Tools

Introduction:

Visual basic uses object oriented techniques to create program that are powerful, robust and efficient.

Start →programs → Microsoft visual studio → Microsoft visual basic 6.0

Project:

Each application in visual basic is called as project. A project is a collection of forms, modules, user controls and data reports etc. it organizes the forms and modules. The project is saved with the extension .vbp.

Forms:

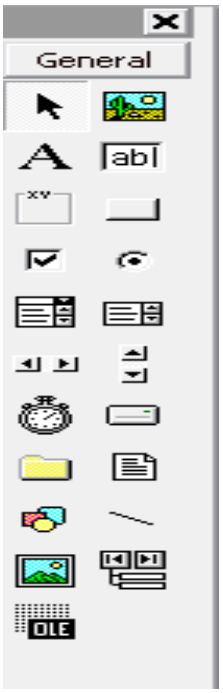
A form is a collection of controls. The controls are placed on the form. The form also has its own properties and methods. It has the extension .frm. - 99 -More than one form may be used in an application.

Visual basic is referred to as an integrated development environment (IDE). IDE consists of following elements,

- Title bar
- Menu bar
- Tool bar
- Tool box
- Control menu
- Project explorer window
- Properties window
- Object browser
- Form designer

- Code editor window
- Form layout window

TOOL BAR:



ADDING AND REMOVING TOOLBAR:

- Right click anywhere on the menu bar, or choose toolbars from the view menu the toolbar pop-up menu appears.
- Select the type of standard toolbar that you want from the pop-up menu. If a check is to the left of the toolbar type, that toolbar is already visible.

Under the menu, there is a toolbar. Toolbar is used to quick access the commonly used menu commands. There are few build in toolbars,

- Standard toolbar
- Edit toolbar
- Debug toolbar
- Form edit toolbar

STANDARD TOOLBAR:

The standard toolbar is the central toolbar in the visual basic IDE. The standard toolbar offers many features found in the file, project, debug and run menu.

The standard toolbar enables fast access to often use functionally and information.

THE EDIT TOOLBAR:

The extended edit menu and some debug menu functions from the edit toolbar can be accessed.

The feature of edit toolbar is similar to those of the edit menu. You can cut, copy and paste text. You can manipulate the layout of the code and do text selection, searches and replacement. Also you can use automatic coding features such as quick info.

THE DEBUG TOOLBAR:

The debug toolbar enables you to access the debugging functions of the visual basic IDE. You can use the debug toolbar to test the program and restore errors that might occur. When you debug a program you do such things as run the code a line at a time.

THE FORM EDITOR TOOLBAR:

You can use the form editor toolbar to size, move, and align controls on a form. The form editor toolbar has the same set of features as the format menu.

You align and size multiple controls on a form with the form editor toolbar. There are small download facing arrowheads to the right of the align, centre and make toolbar buttons. These arrowheads indicate that a dropdown menu will appear when you select that toolbar button.

PROJECT EXPLORER WINDOW:

To experiment with the project explorer window, click the toggle folders button. Notice that the folders are collapsed.

To expand the folders, click the toggle button again. Still on the project explorer window, click view code. You are presented with the code editor window.

To send the code editor window to the background again, on the project explorer windows, click the view object button.

DEFAULT CONTROLS:

Common properties:

Important common properties include the following,

- Name
- Index
- Left
- Top
- Height
- Width
- Enabled
- Visible

Controls contained in the visual basic toolbox:

1) **Picture box**: Displays graphics. Can also serve as a container for other controls.
   Property: caption, picture.
2) **Label box**: Displays text that user cannot edit.
   Property: caption.
3) **Text box:** Displays text. Allows the user to enter and edit text.
   Property: text.
4) **Frame**: Serves as a container for other commands. Provides grouping of controls.
   Property: caption.
5) **Command buttons:** Allows the user to initiate actions by clicking the button.
6) **Check box**: Lets the user make a true/false choice.
7) **Option button:** Lets the users choose from one option from a group of items.
   Property: caption.
8) **Combo box**: Lets the users choose from a list of items or enter a new values.
   Property: caption.
9) **List box**: Lets the user choose from a list of items.
   Property: list.
10) **Horizontal/ Vertical scroll box**: Lets the user choose a scrollbar value based on the position of button in the bar.

11) **Timer**: Lets the program perform functions on a timed basic.
12) **Drive list box:** Let the user select a disk drive.
13) **Directory list**: Let the users select a box directory or folders.
14) **File list box**: Lets the user select a file.
15) **Shape**: Displays a shape on the form.
16) **Line:** Displays a line on the form.
17) **Image**: similar to a picture box control, uses fewer system resources but doesn't support as many properties, events and methods.
18) **Data control**: Provides an interface between the program and a data source.
19) **OLE**: Provides a connection between the program and an OLE server.
20) **Common dialog**: Allows use of windows standard dialog boxes to retrieve information such as filenames and colors.

RESULT:

Thus, front end tool is executed.

EX.NO:4

FORMS-TRIGGERS-MENU DESIGN

Aim:

To study and execute Triggers in RDBMS.

Definition & Syntax: -

TRIGGER:

A database trigger is a stored procedure that is fired when an insert, update or delete statement is issued against the associated table. Database triggers can be used for the following purposes.

To generate data automatically.

To enforce complex integrity constraints. (e.g., Checking with sysdate, checking with data in another table).

To customize complex security authorizations.

To maintain replicate tables.

To audit data modifications.

Syntax for Creating Triggers

The syntax for creating a trigger is given below

```
CREATE OR REPLACE TRIGGER <trigger_name>
[BEFORE/AFTER] [INSERT/UPDATE/DELETE] ON <table_name>
[FOR EACH statement/FOR EACH row]
[WHEN <condition>]
PL/SQL block;
```

PARTS OF A TRIGGER

A database trigger has three parts, namely, a trigger statement, a trigger body and a trigger restriction.

TRIGGER STATEMENT:

A trigger statement specifies the DML statements like update, delete and insert and it fires the trigger body. It also specifies the table to which the trigger is associated.

TRIGGER BODY:

Trigger body is a PL/SQL block that is executed when a triggering statement is issued.

TRIGGER RESTRICTION:

Restrictions on a trigger can be achieved using the WHEN clause as shown in the syntax for creating triggers. They can be included in the definition of a row trigger, where in, the condition in the WHEN clause is evaluated for each row that is affected by the trigger.

TYPES OF TRIGGER:

Triggers are categorized into the following types based on when they are fired:

- Before
- After
- For each row
- For each statement (default)

BEFORE /AFTER OPTIONS:

The before/after options can be used to specify when the trigger body should be fired with respect to the triggering statement. If the user includes a BEFORE option, then, Oracle fires the trigger before executing the triggering statement. On the other hand, if AFTER is used, then, Oracle fires the trigger after executing the triggering statement.

FOR EACH ROW / STATEMENT:

When the for each row / statement option when included in the 'create trigger' syntax specifies that the trigger fires once per row. By default, a database trigger fires for each statement.

Using a combination of the above options, we can assign 12 types of triggers to a database table.

Before update row / statement

Before delete row / statement

Before insert row / statement

After update row / statement

After delete row / statement

After insert row / statement

EXERCISE:

1. Write a PL/SQL program to create a trigger before the user inserts the data into the table.
2. Write a PL/SQL program to create a trigger before the user deletes the data from the table.
3. Write a PL/SQL program to create a trigger before the user changes the value of the salary of the employee.

ANSWERS:

```
SQL>create or replace trigger ins1 before insert on emp begin
raise_application_error (-20001,'you can’t insert a row'); end;
```

OUTPUT:

```
SQL>insert into emp
values(&eid,'&name','&dob','&addr','&sex','&desig',&deptno,'&maritsta',&salary);
SQL>insert into emp *
values(&eid,'&name','&dob','&addr','&sex','&desig',&deptno,'&maritsta',&salary);
 ERROR at line 1:
ORA-20001: you cant insert a row
ORA-06512: at "CSE382.ins1", line 2
ORA-04088: error during execution of trigger 'CSE382.ins1'
```

```
SQL>create or replace trigger del1 before delete on emp
begin
raise_application_error (-20001,'you can’t delete a row');
end;
```

OUTPUT:

```
SQL>delete from emp where eid=4444;
delete from emp where eid=4444;
                  *
ORA-20001: you can't delete a row
ORA-06512: at "CSE382.DEL1", line 2
ORA-04088: error during execution of trigger 'CSE382.DEL1'

SQL> create trigger upd1 before update on emp for each row 2  begin
  3  if :new.sal < 1000 then
  4  raise_application_error(-20001,'salary can't be low thanthis'); 5 end if;
  6  end;
  7  /
Trigger created.

SQL> update emp set sal=500 where dno=2;
update emp set sal=500 where dno=2
       *
ERROR at line 1:
ORA-20001: salary can't be low than this
ORA-06512: at "CSE382.UPD1", line 3
ORA-04088: error during execution of trigger 'CSE382.UPD1'
```

RESULT:

Thus forms-triggers-menu design is executed.

EX.NO: 5

REPORTS

Aim:

To design generate reports by using VB and oracle.

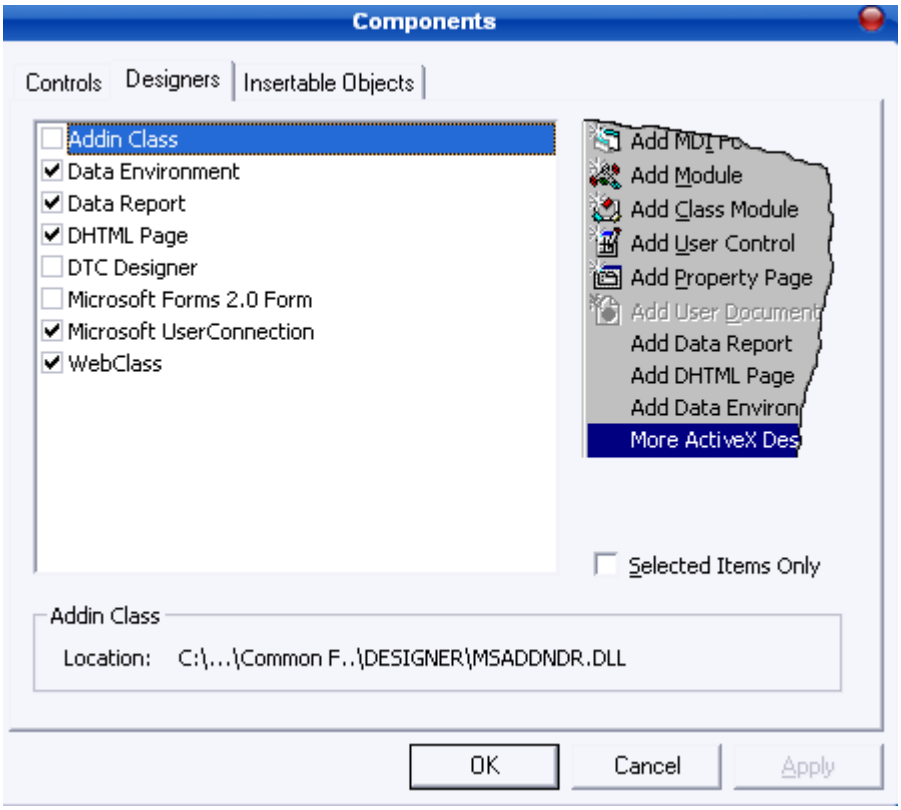
Steps:

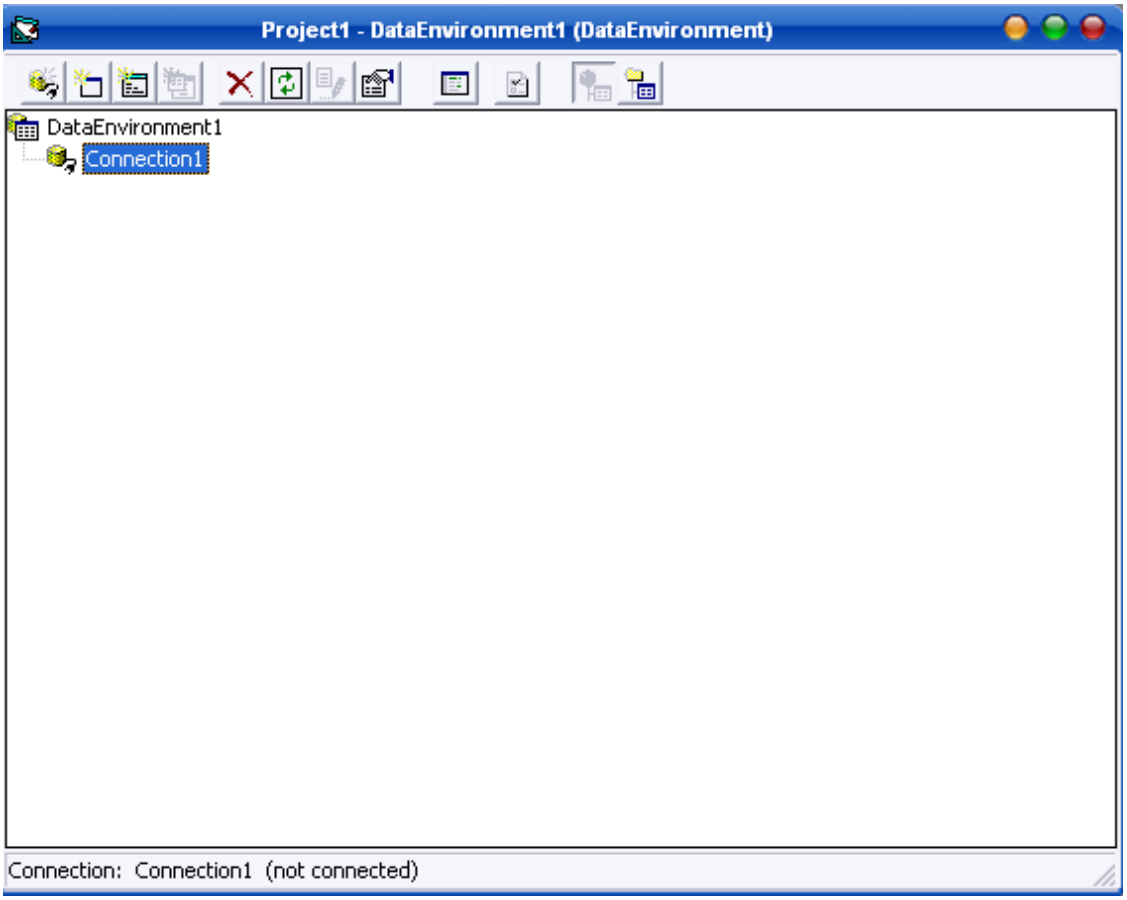
1. Project ✇Components ✇Designer tab✇ Check the following
   - Data Report
   - Data Environment
2. Project Explorer Right click – Add Data Environment.
3. Click connection1 properties ✇ MS OLEDB provider for oracle✇ Click next ✇ type the username and password in data link properties ✇ click test connection and it will display the message if the connection is true.
4. Right click the connectio1 in data environment add command option click command1 in connection1 ✇Go to properties ✇To enter database object table and select the object name.
5. Project Explorer ✇ Right click –Add data report in data report properties. Set the following properties
   - Data source ✇ Give data environment name.
   - Data member ✇ Give command name.
6. Drag the command1 object in data report in detail section.
7. Arrange the title in page header section and design the data report in specified section.
8. Create one form with one command button name in show report .
   Button click event

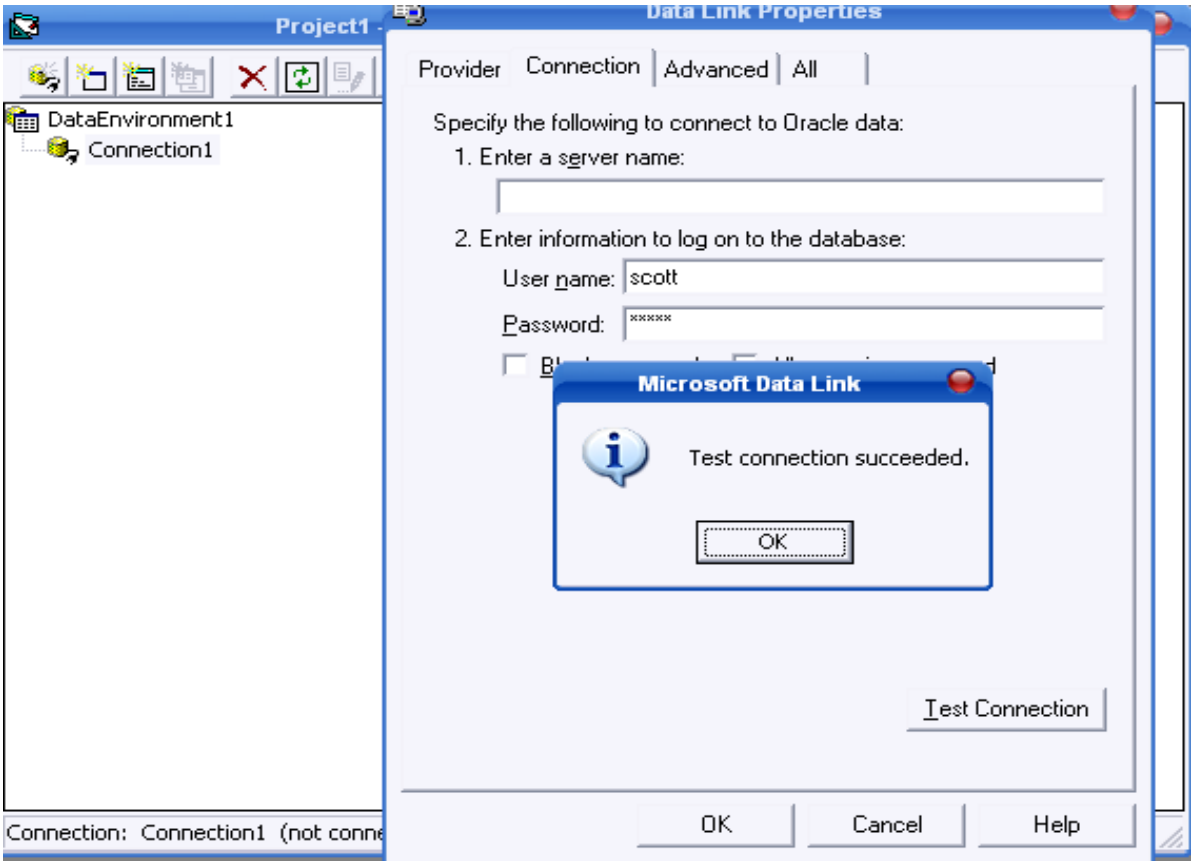
   Report name.Show

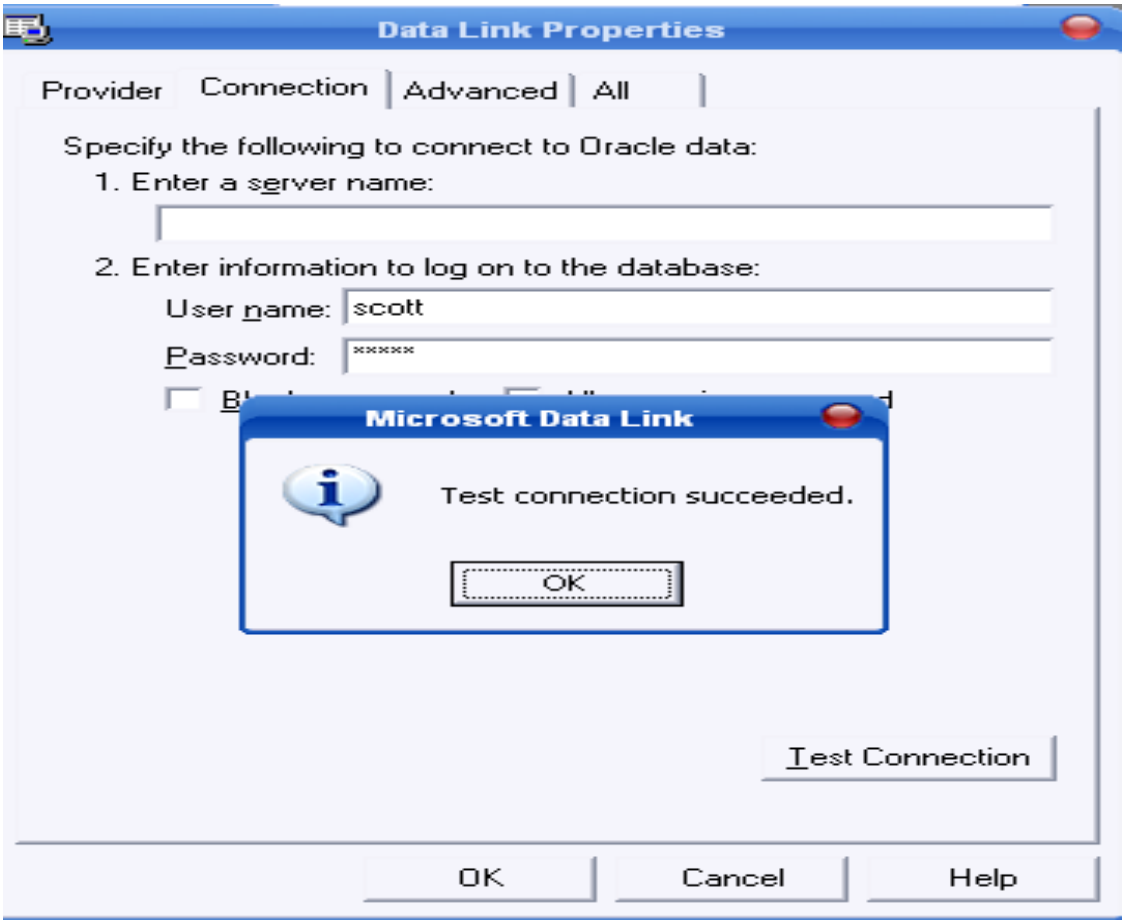
   Eg

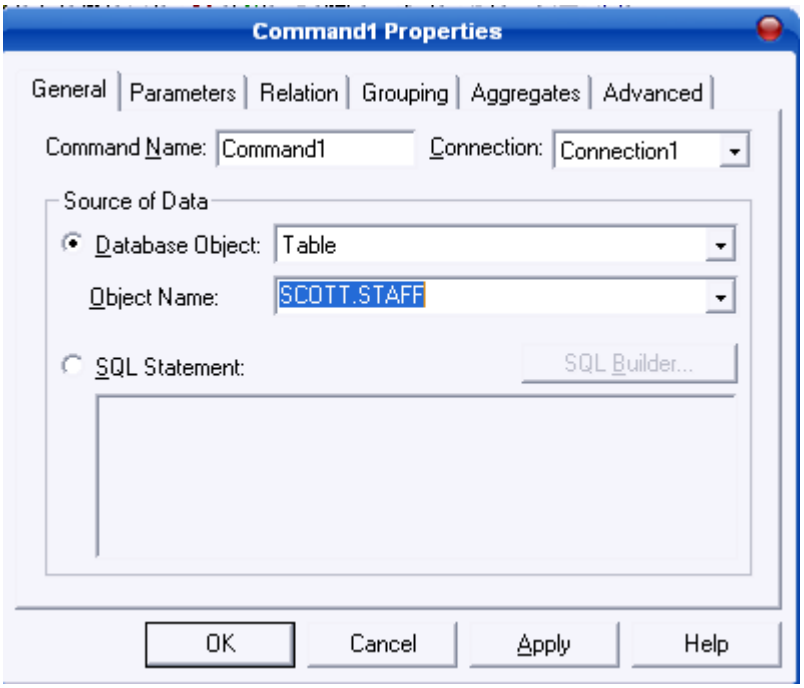
   Data report1.Show

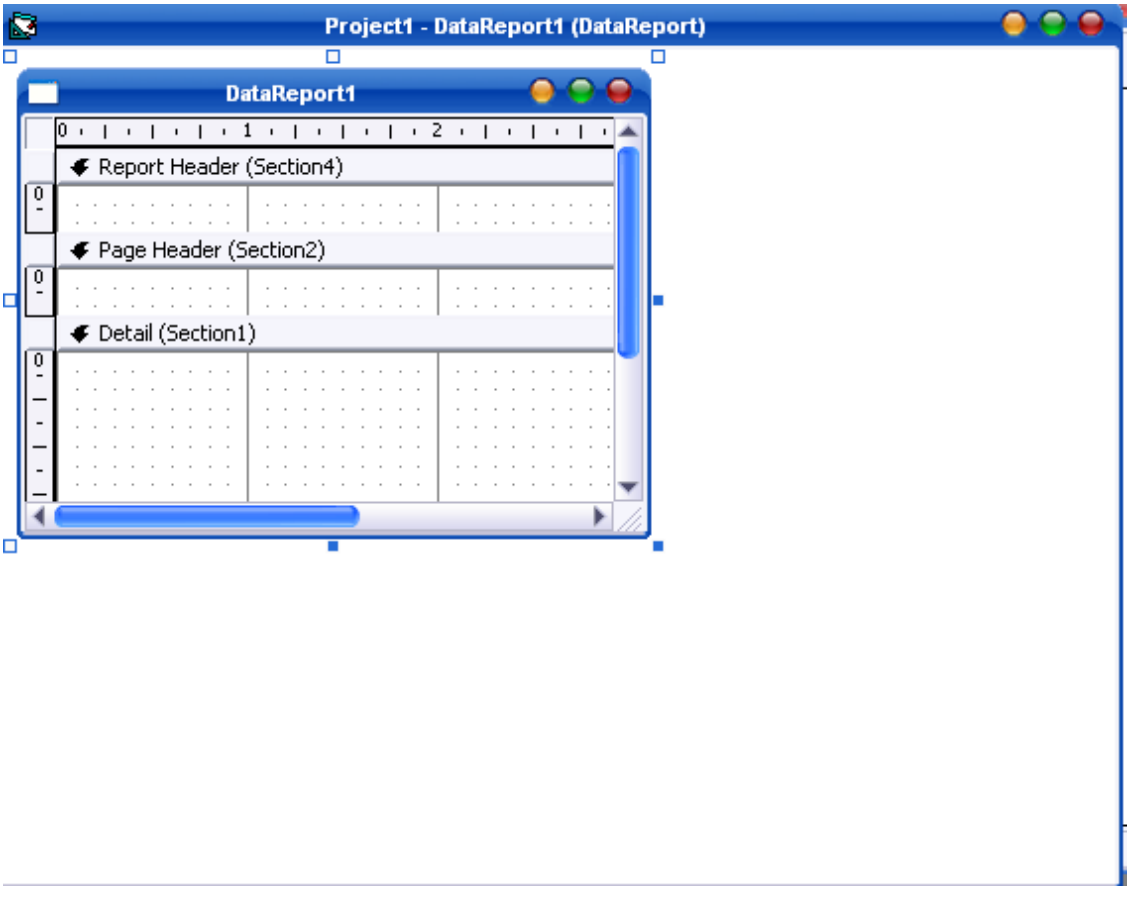

Components
Controls
Designers
Insertable Objects
Addin Class
Data Environment
Data Report
DHTML Page
DTC Designer
Microsoft Forms 2.0 Form
Microsoft UserConnection
WebClass
Add MDI Fo
Add Module
Add Class Module
Add User Control
Add Property Page
Add User Document
Add Data Report
Add DHTML Page
Add Data Environ
More ActiveX Des
Selected Items Only
Addin Class
Location: C:\...\Common F..\DESIGNER\MSADDNDR.DLL
OK
Cancel
Apply

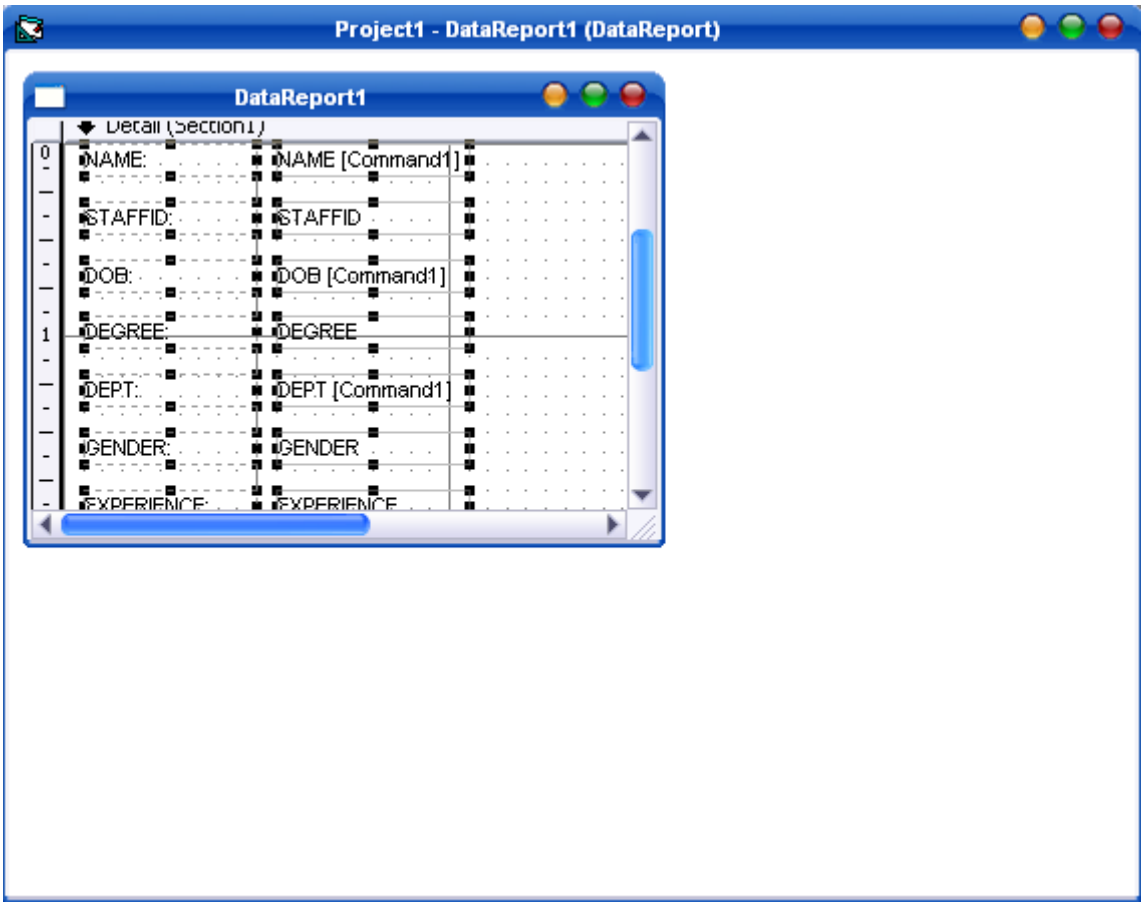

Project1 - DataEnvironment1 (DataEnvironment)
DataEnvironment1
Connection1
Connection: Connection1 (not connected)

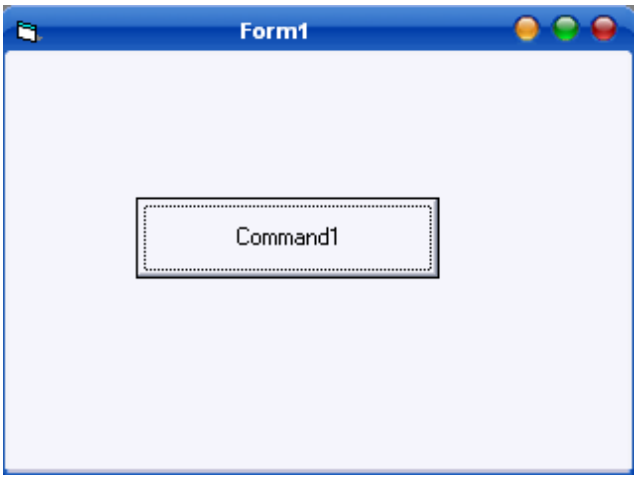

Project1
DataEnvironment1
Connection1
Connection: Connection1 (not conne
Data Link Properties
Provider
Connection
Advanced
All
Specify the following to connect to Oracle data:
1. Enter a server name:
2. Enter information to log on to the database:
User name:
scott
Password:
*****
Microsoft Data Link
Test connection succeeded.
OK
Test Connection
OK
Cancel
Help

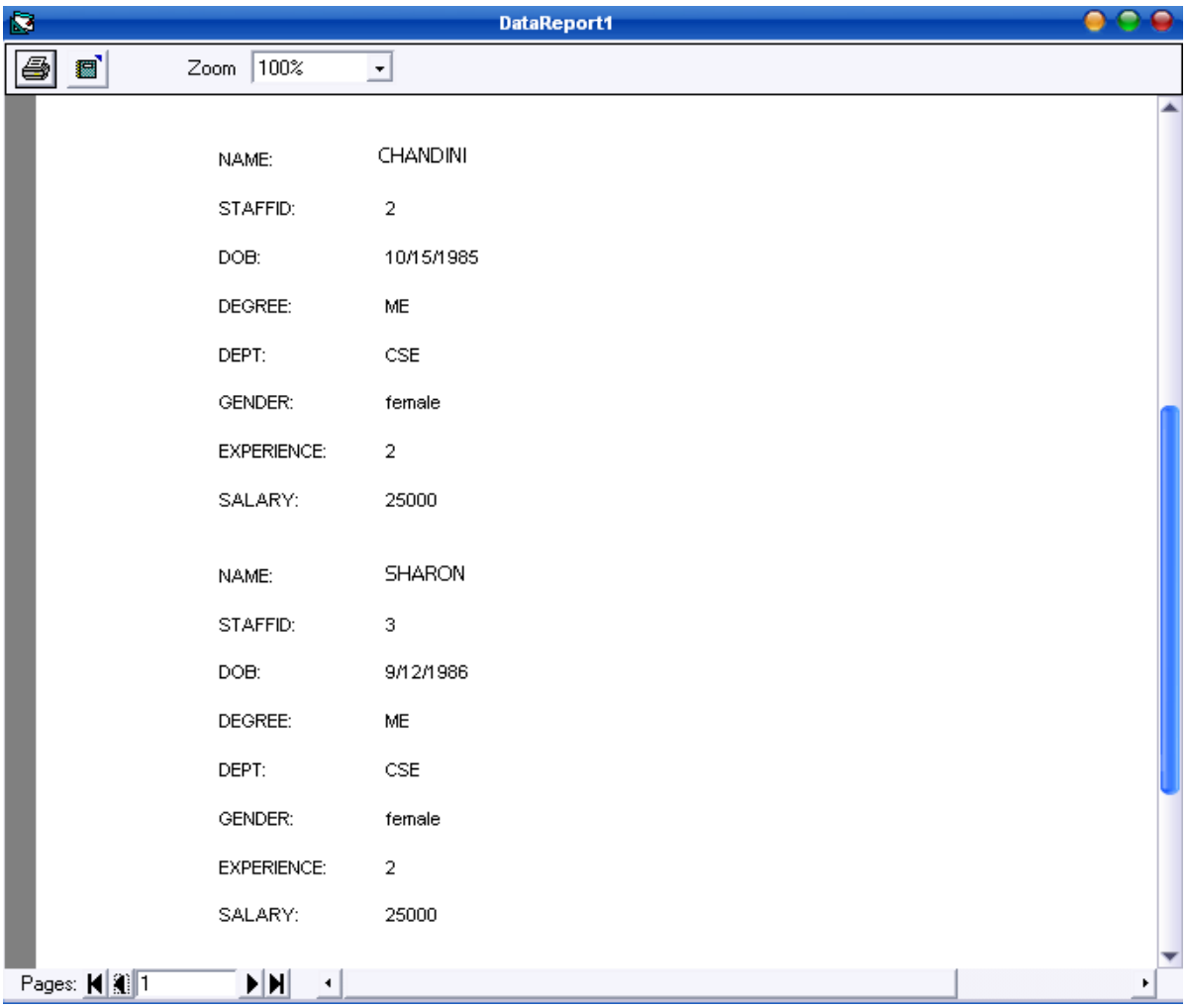

Data Link Properties
Provider
Connection
Advanced
All
Specify the following to connect to Oracle data:
1. Enter a server name:
2. Enter information to log on to the database:
User name:
scott
Password:
******
Microsoft Data Link
Test connection succeeded.
OK
Test Connection
OK
Cancel
Help


Command1 Properties
General
Parameters
Relation
Grouping
Aggregates
Advanced
Command Name:
Command1
Connection:
Connection1
Source of Data
Database Object:
Table
Object Name:
SCOTT.STAFF
SQL Statement:
SQL Builder...
OK
Cancel
Apply
Help


Project1 - DataReport1 (DataReport)
DataReport1
Report Header (Section4)
Page Header (Section2)
Detail (Section1)


Project1 - DataReport1 (DataReport)
DataReport1
Detail (Section1)
NAME:
NAME [Command1]
STAFFID:
STAFFID
DOB:
DOB [Command1]
DEGREE:
DEGREE
DEPT:
DEPT [Command1]
GENDER:
GENDER
EXPERIENCE:
EXPERIENCE


Form1
Command1



CRITERIA	MAX.MARKS	MARKSOBTAINED
AIM& ALGORITHM	**5**	
EXECUTION&OUTPUT	**10**	
VIVA	**5**	
TOTAL	**20**	

RESULT:

Thus, My SQL program for reports is executed

EX.NO: 6

DESIGN AND IMPLEMENTATION OF EMPLOYEE

DATABASE IN BANK

Aim:

To design a forms and write a code for banking systems and make a connection with back end using ADO Data control.

Table Used: Employee:

NAME	FATHER _NAME	EMP. _No	DOB	SEX	MOTHER TONGUE	CITY	STREET	STATE
Sekar	Moorthy	101	2/2/80	Male	Hindi	Delhi	Clive St	Delhi
Ajith	Arjun	102	23/9/81	Male	English	Banglore	MG St	KA
Anitha	Arun	103	30/10/75	Female	Tamil	Chennai	KKnagar	TN
Kowsi	Maridass	104	20/1/87	Female	Telugu	Hydrabad	Port st	AP

Description:

Table creation:

The student database has been created in Oracle and some rows have been inserted using the DDL and DML command.

Table Creation:

SQL> create table employee (name varchar2(20), f_name varchar2(15), emp_no number(5), dob date, sex varchar2(5) , m_tong varchar2(10) , city varchar2(10) , street varchar2(10) , state varchar2(10));

Table created.

Values Are Inserted By

SQL>Insert into employee values (‘&name’, ‘&f_name’, &emp_no, ‘&dob’, ‘&sex’, ‘&m_tong’, ‘&city’, ‘&street’, ‘&state’);

To open visual basic:

1) Go to start → all programs→ Microsoft Visual Studio 6.0 → Microsoft Visual Basic 6.0
   →Click.

To open a new form:

2) While opening it will ask you New project in that click standard exe→ then open→ new
   Form is opened. (Or)

   Go to File menu →click new project→new form is opened.

To bring the toolbar:

3) Go to tools menu→click toolbar→tool bar is loaded.

To create a form:

4) From the tool bar drag the text box and label and place it in the form.
5) The number of text box and label depends upon the fields we have in the Table.
6) We can also have command buttons to perform particular action when they are clicked.
7) To view the form we should press shift +F7.

Data control:

Visual Basic provides a set of controls that allow you to display, add edit data in the database with minimal coding. When such controls are used the user need not write code, instead they allow the user to use their properties to access the database. Such controls are known as Data-aware controls. Data controls are a standard control available in the tool box.

Let us consider that we are maintaining a database named emp, which consists of fields like empno, empname, empadd, empphone. The steps to connect the data control to the emp database are:

1. Place the Data control on the form by double clicking on the icon representing the Data control in the toolbox.

2. Place four text boxes on the form to display the value of the fields empno, empname, empadd and empphone from the table emp into respectively.

3. Set the connection string property of the Data control to Access. The Connection string property determines the type of the database to access.

4. Set the DatabaseName property to emp. The DatabaseName property determines the name of the database to be opened.

5. Set the RecordSource property of the Data control to empinfo. The RecordSource property determines the name of the table to be accessed.

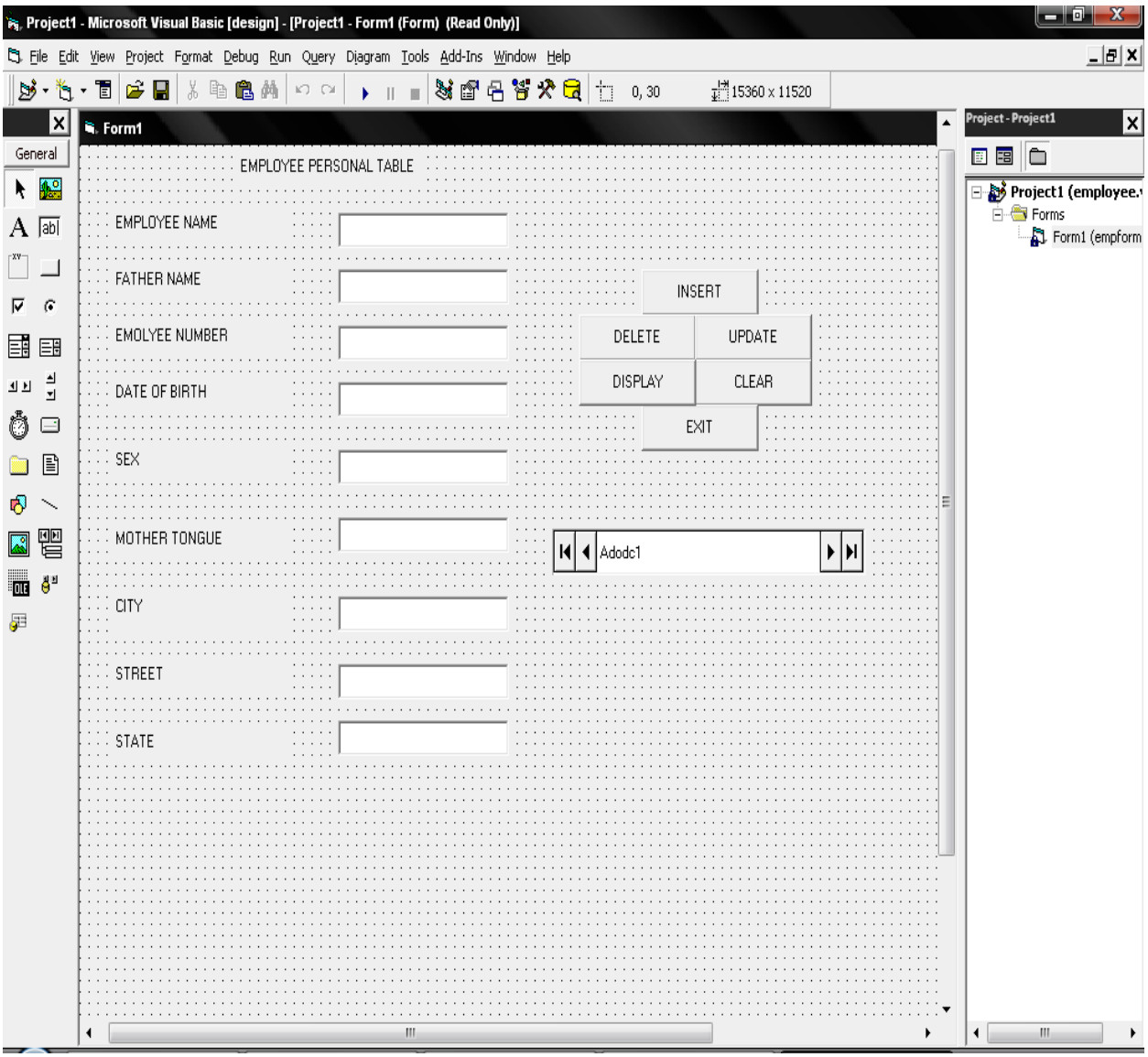
6. Make the text boxes bound to the Data control by using the DataSource and DataField properties. A control is said to be data-aware when it is bound to a Data control. The DataSource property determines the name of the Data control to which the text box is to be bound. The DataField property determines the name of the field in the table. Set the Name, DataSource and DataField properties of the text boxes as shown in the below table.

Object	**Property**	**Setting**
Text1	Name	txtempno
	DataSource	data 1
	DataField	empno
ADODB1	ConnectionString	Provider=MSDAORA.1; User ID=scott;Persist Security Info=false
	Password	tiger

	RecordSource	empinfo
	UserName	scott

7. Run the application and use the arrow buttons on the data control to navigate through the records in the below screen. You have to write code to add, update, edit and delete records in the below screen. Or else press the function key 5 (F5)

Form Design:



CODING WINDOW:

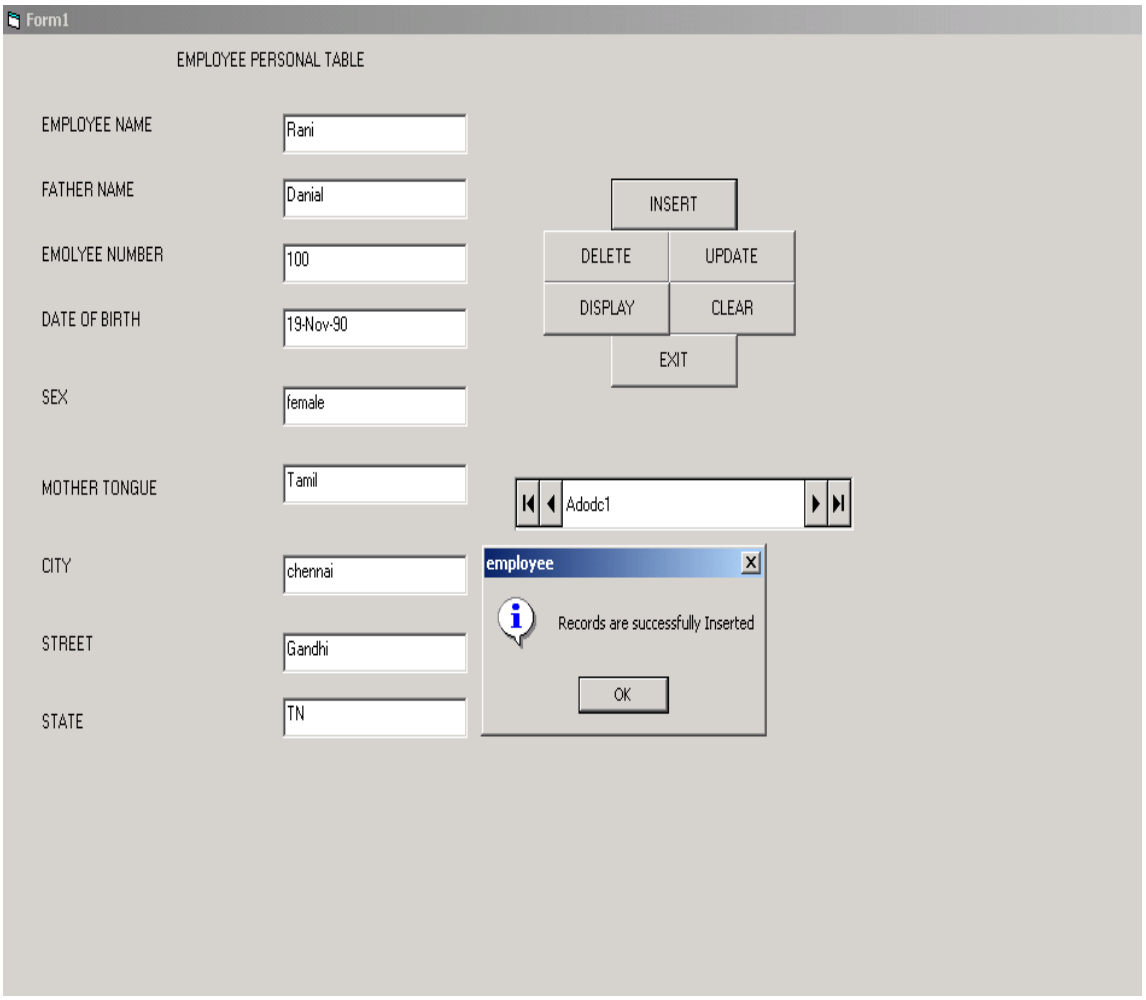
```
Private Sub CLEAR_Click ()
    Text1.Text = ""
    Text2.Text = ""
    Text3.Text = ""
    Text4.Text = ""
    Text5.Text = ""
    Text6.Text = ""
    Text7.Text = ""
    Text8.Text = ""
    Text9.Text = ""
End Sub

Private Sub DELETE_Click()
    Adodc1.Recordset.DELETE
    MsgBox "Records are Deleted successfully", vbInformation
End Sub

Private Sub INSERT_Click()
    Adodc1.Recordset.AddNew
   MsgBox "Records are Inserted successfully", vbInformation
End Sub

Private Sub UPDATE_Click()
    Adodc1.Recordset.UPDATE
    MsgBox "Records are Updated successfully", vbInformation
End Sub

Private Sub DISPLAY_Click()
    rs.Open " select * from bank where acc_no=" & Text1.Text & " ", Con, adOpenStatic
    If rs.BOF Then
       MsgBox "No Such Record Found", vbInformation
    Else
```

```
    MsgBox "Record Found", vbInformation
    Text1.Text = rs.Fields("name")
    Text2.Text = rs.Fields("f_name")
    Text3.Text = rs.Fields("emp_n0")
    Text4.Text = rs.Fields("dob")
    Text5.Text = rs.Fields("sex")
    Text6.Text = rs.Fields("m_tong")
    Text7.Text = rs.Fields("city")
    Text8.Text = rs.Fields("street")
    Text9.Text = rs.Fields("state")
  End If
```

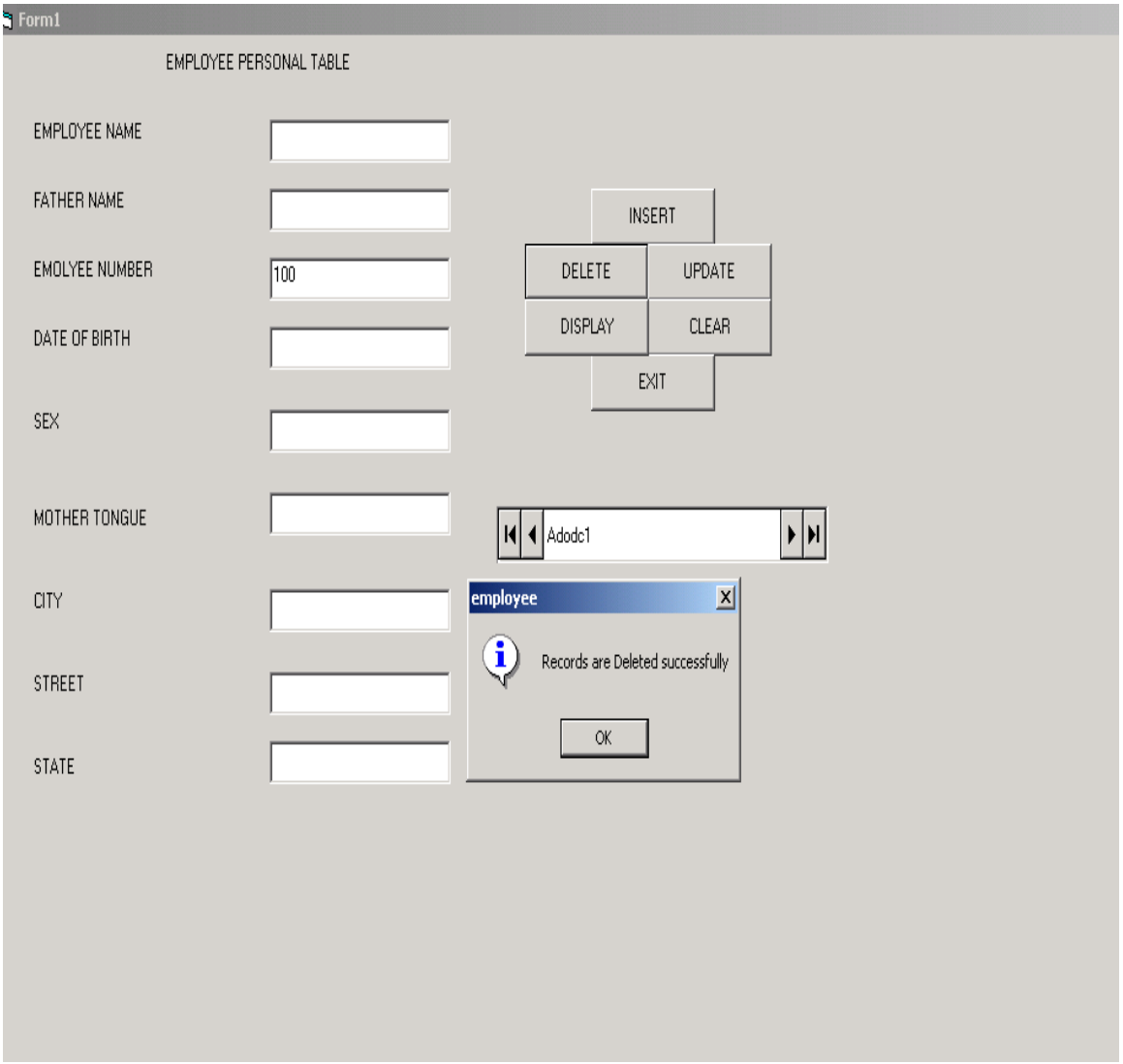
End Sub

Screen Shots:

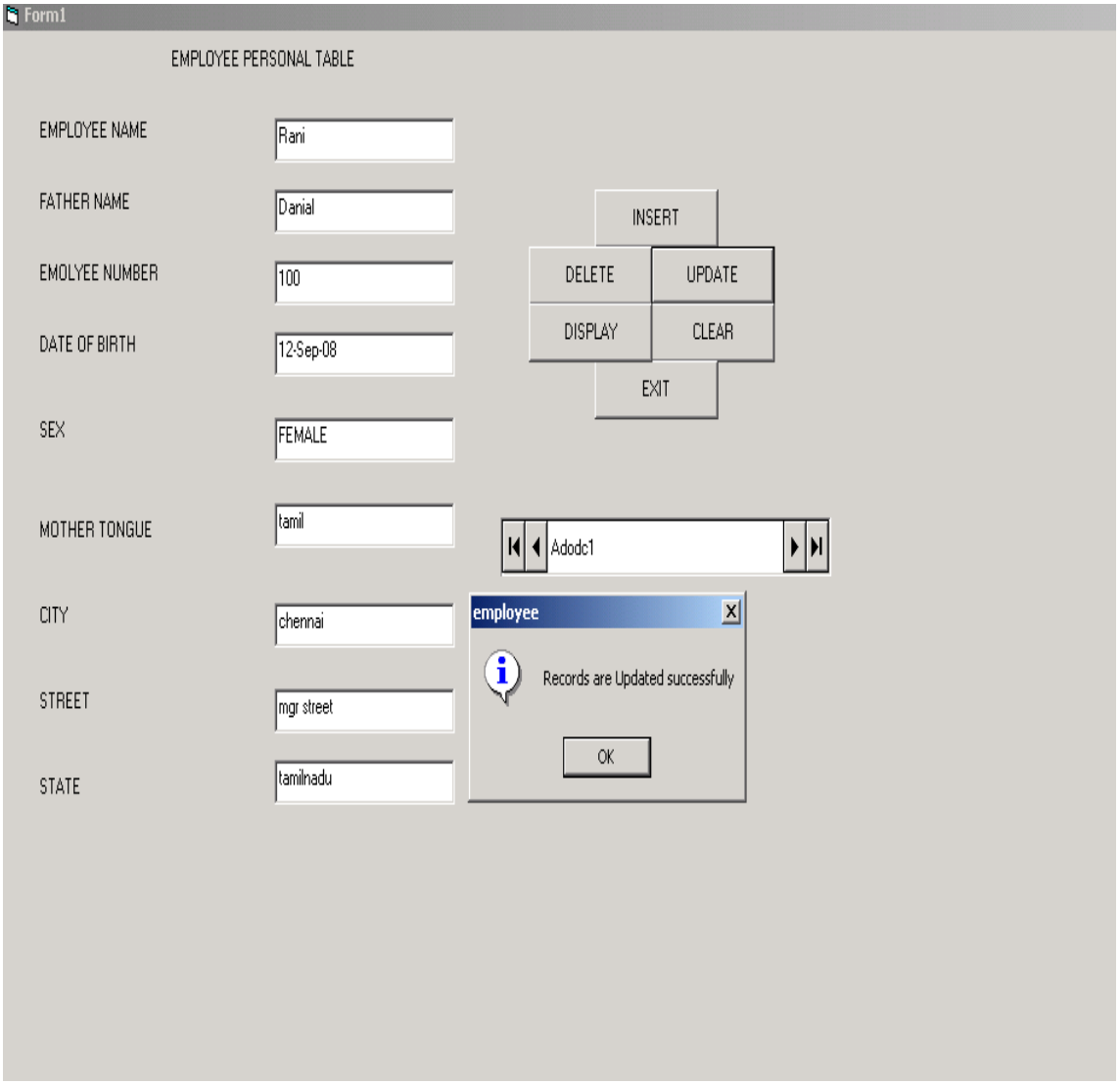
1. **Insertion**



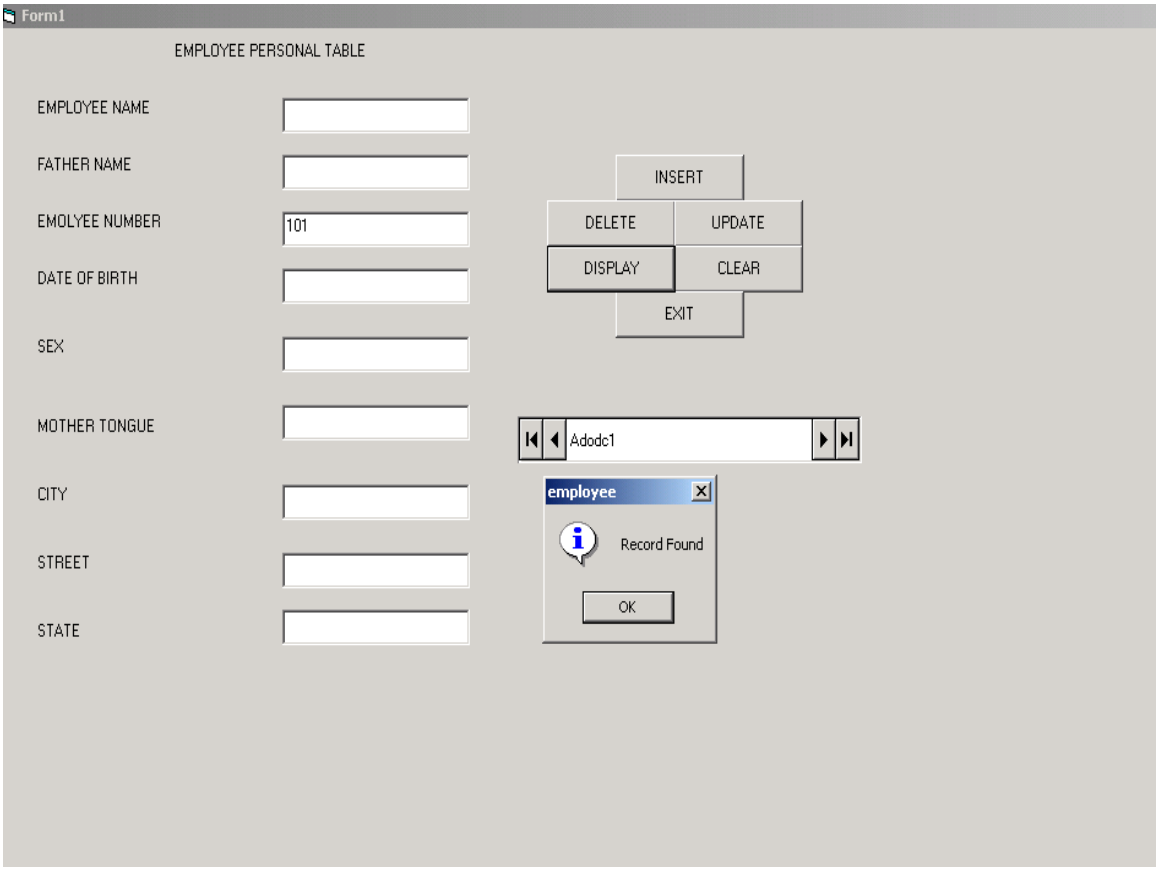
1. Deleting


Form1
EMPLOYEE PERSONAL TABLE
EMPLOYEE NAME
FATHER NAME
EMOLYEE NUMBER
100
DATE OF BIRTH
SEX
MOTHER TONGUE
CITY
STREET
STATE
INSERT
DELETE
UPDATE
DISPLAY
CLEAR
EXIT
Adodc1
employee
Records are Deleted successfully
OK

2. Updating


Form1
EMPLOYEE PERSONAL TABLE
EMPLOYEE NAME
Rani
FATHER NAME
Danial
EMOLYEE NUMBER
100
DATE OF BIRTH
12-Sep-08
SEX
FEMALE
MOTHER TONGUE
tamil
CITY
chennai
STREET
mgr street
STATE
tamilnadu
INSERT
DELETE
UPDATE
DISPLAY
CLEAR
EXIT
Adodc1
employee
Records are Updated successfully
OK

3. Display a particular record



RESULT:

Thus design and implementation of employee is executed.

EX.NO: 7

An Exercise using Open-Source Software like My SQL

Aim:

To use an open-Source Software My SQL and create a simple table with countries including columns with country-id, country-name and region-id.

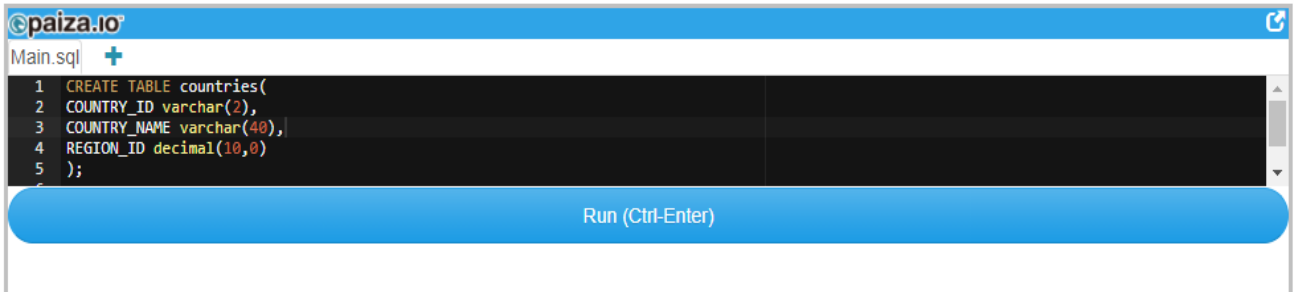
Code:

```
CREATE TABLE countries (
COUNTRY_ID varchar(2),
COUNTRY_NAME varchar(40),
REGION_ID decimal(10,0)
);
```

Execute the above code in MySQL 5.6 command prompt

Output :

```
mysql> DESC countries;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| COUNTRY_ID   | varchar(2)    | YES  |     | NULL    |       |
| COUNTRY_NAME | varchar(40)   | YES  |     | NULL    |       |
| REGION_ID    | decimal(10,0) | YES  |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+

3 rows in set (0.01 sec)
```



RESULT:

Thus, open-Source Software My SQL was used to create a simple table.